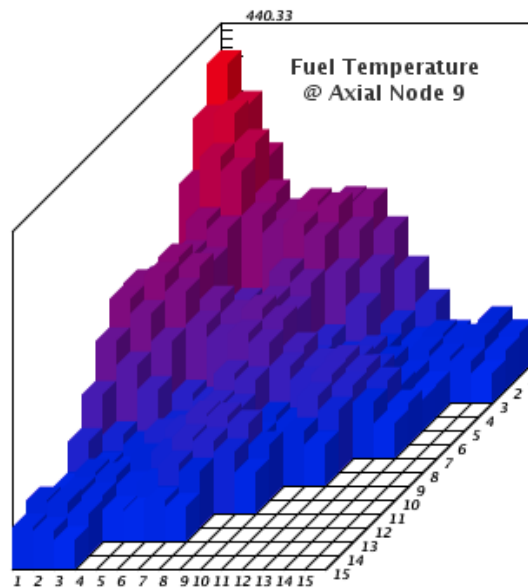


# Symbolic Nuclear Analysis Package (SNAP)

## User's Manual



April 2007

Applied Programming Technology, Inc.

240 Market St., Suite 208  
Bloomsburg PA 17815-1951

---

# Symbolic Nuclear Analysis Package (SNAP) User's Manual

Applied Programming Technology, Inc.

Copyright © 2006-2007

\*\*\*\*\* Disclaimer of Liability Notice \*\*\*\*\*

The Nuclear Regulatory Commission and Applied Programming Technology, Inc. provide no express warranties and/or guarantees and further disclaims all other warranties of any kind whether statutory, written, oral, or implied as to the quality, character, or description of products and services, its merchantability, or its fitness for any use or purpose. Further, no warranties are given that products and services shall be error free or that they shall operate on specific hardware configurations. In no event shall the US Nuclear Regulatory Commission or Applied Programming Technology, Inc. be liable, whether foreseeable or unforeseeable, for direct, incidental, indirect, special, or consequential damages, including but not limited to loss of use, loss of profit, loss of data, data being rendered inaccurate, liabilities or penalties incurred by any party, or losses sustained by third parties even if the Nuclear Regulatory Commission or Applied Programming Technology, Inc. have been advised of the possibilities of such damages or losses.

---

# Table of Contents

1. Introduction .....	1
2. The ModelEditor .....	3
2.1. The ModelEditor User Interface .....	3
2.1.1. The Main Toolbars .....	4
2.1.2. Main Menu Items .....	4
2.1.3. The Navigator .....	5
2.1.4. The Main Property View .....	6
2.1.5. 2D Views .....	7
2.1.6. The Message Window .....	13
2.2. User Preferences .....	13
2.2.1. General Preferences .....	14
2.2.2. Color .....	16
2.2.3. Window Arrangement .....	16
2.3. Creating a Model .....	16
2.3.1. Opening a New Model .....	16
2.3.2. Creating and Editing Components .....	18
2.3.3. Creating Annotations .....	20
2.3.4. Component Display Scaling .....	25
2.3.5. Checking a Model for Errors .....	25
2.3.6. Saving a Model .....	26
2.4. User Defined Numerics .....	26
2.4.1. Constants and Variables .....	26
2.4.2. Numerics in a View .....	28
2.4.3. User Defined Functions .....	29
2.5. Command Line Usage .....	30
2.6. Batch Command Syntax .....	31
3. The Configuration Tool .....	35
3.1. User Interface .....	35
3.1.1. Property View .....	36
3.1.2. Menu Items .....	36
3.2. Configuration Tool Properties .....	36
3.2.1. General Properties .....	36
3.2.2. Calculation Server Properties .....	37
3.2.3. Web Support .....	40
4. Job Status .....	41
4.1. User Interface .....	41
4.2. Menu Items .....	43
4.3. Root Folders .....	44
4.4. Adding a Remote Server .....	45
4.5. Importing a Local File .....	45
5. Submitting a Job .....	47
6. The Calculation Server .....	51
7. The Animation Plug-in .....	53
7.1. Creating a New Animation Model .....	53
7.2. Animation Components .....	54
7.3. Data Sources .....	54
7.4. Sequenced Data Sources .....	55
7.5. The Python Data Source .....	55

7.6. Ranges .....	57
7.7. Animation Playback .....	58
7.8. Display Beans .....	59
7.8.1. Control System .....	62
7.8.2. Indicators .....	63
7.8.3. Interactive .....	68
7.8.4. Plant Components .....	69
8. Using the jEdit Plug-in .....	75
A. Installing the jEdit Plug-in for SNAP .....	77
A.1. Installing jEdit .....	77
A.2. Installing the Plug-in Files .....	77
A.3. Update the jEdit Configuration .....	77
A.4. Set the SNAP Installation Directory .....	78
A.5. Set the jEdit Executable .....	78
B. Calculation Job Files .....	79
B.1. A Sample Calculation Job File .....	79
Index .....	81



---

# Chapter 1. Introduction

The Symbolic Nuclear Analysis Package (SNAP) consists of a suite of integrated applications designed to simplify the process of performing engineering analysis. SNAP is built on the Common Application Framework for Engineering Analysis (CAFEAN) which provides a highly flexible framework for creating and editing input for engineering analysis codes as well as extensive functionality for submitting, monitoring, and interacting with the codes.

SNAP currently includes support the CONTAIN, COBRA, FRAPCON-3, MELCOR, PARCS, RELAP5 and TRACE analysis codes. Each code is supported by a separate plug-in. The list of currently installed plug-ins can be found in the [Chapter 2, The ModelEditor](#) **About** dialog by pressing the **Plugins** button.

## The SNAP Application Suite

The SNAP application suite includes the ModelEditor, JobStatus and the Configuration Tool client applications as well as the Calculation Server. In this context, a client application is one that typically is run on the local machine and provides a graphical user interface. A server application is one that runs in the background or on a different computer to provide access to data.

The ModelEditor is the primary SNAP client-side user interface. It is responsible for the development and modification of input models for the supported analysis codes. It is also responsible for animating the results of those analyses using the Animation plug-in. The Configuration Tool is used to configure properties for the SNAP client applications as well as to startup, shutdown and configure the Calculation Server. Job Status is used to display the status of jobs on a server as well as to create new jobs by importing local data files. The Calculation Server provides control of and communication with active and completed calculations.

For more information on the SNAP applications refer to: [Chapter 2, The ModelEditor](#), [Chapter 4, Job Status](#), [Chapter 3, The Configuration Tool](#), [Chapter 6, The Calculation Server](#), [Chapter 7, The Animation Plug-in](#)

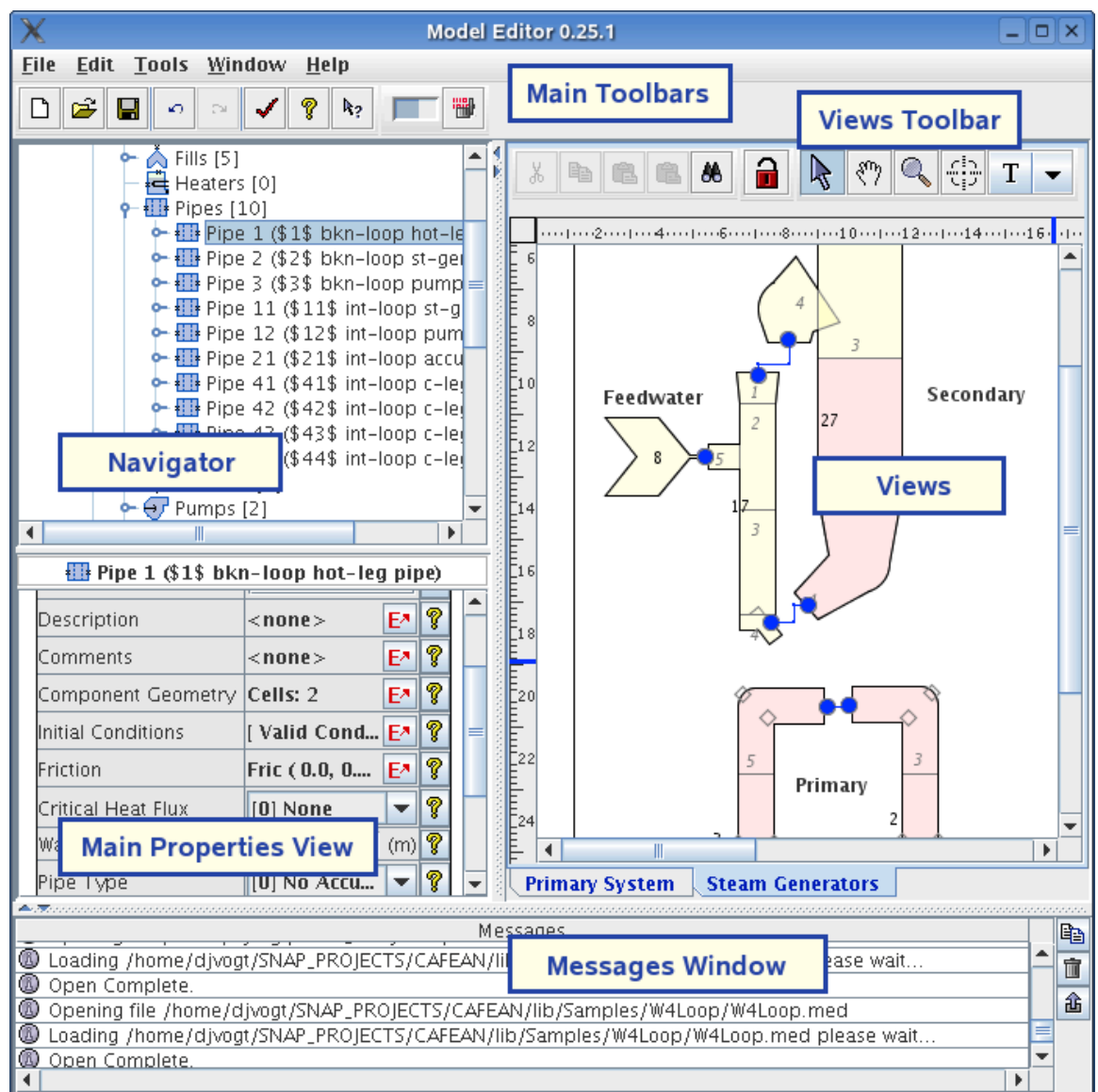
---

# Chapter 2. The ModelEditor

The ModelEditor is the primary SNAP client application. With it, a user may both design input for analysis codes and display animated representations of calculation results. The ModelEditor provides a consistent interface regardless of the analysis code in question.

## 2.1. The ModelEditor User Interface

SNAP's ModelEditor user interface is illustrated in [Figure 2.1, “ModelEditor UI”](#). The primary components are labeled and a brief description of each is provided below.



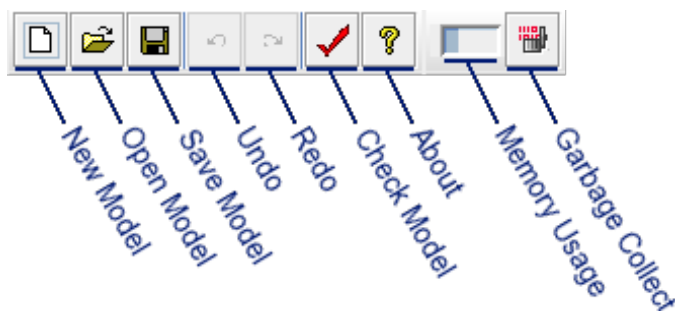
**Figure 2.1. ModelEditor UI**

## 2.1.1. The Main Toolbars

The main toolbars include basic file operations, model operations, and undo/redo (see [Figure 2.2, “ModelEditor Main Toolbar”](#)). These toolbars can be individually enabled or disabled using a right-click pop-up menu located on the main toolbar panel.

Buttons for the basic file and model operations include **New**, **Open** and **Save** as well as shortcuts for **Undo/Redo**, **Check Model**, and **About**.

The memory toolbar displays information on the current memory usage.



*Figure 2.2. ModelEditor Main Toolbar*

## 2.1.2. Main Menu Items

- **File** - The **File** menu contains a standard list of options to create, open, save and close a model, as well as a list of recently opened files in the **Open Recent** menu, and an **Exit** option. The **File** menu also contains two important sub-menus: **Import** and **Export**.

The **Import** menu contains a menu of the file formats supported by the currently loaded plug-ins. In the case of COBRA, the **COBRA ASCII** item will create a new COBRA model and import the given file as COBRA input.

Similarly, the **Export** menu contains a menu of the export file formats supported by the currently selected model.

- **Edit** - This menu contains menu items for the **Undo/Redo** operations and an item to open the Preferences Dialog. The Edit menu also contains an item for the **Plugin Manager** which can be used to disable unused plug-ins.
- The **Tools** menu contains the **Check Model**, **Submit Job** and **Export to jEdit** items.

**Check Model** - The Check Model button and menu item both perform a model validation check on the currently selected model. The checks performed for a model are defined by its plug-in.

**Submit Job** - The current model can be submitted to the Calculation Server by selecting the **Submit Job** menu item from the main **Tools** menu. Refer to [Chapter 5, Submitting a Job](#) for more information on the job submission process.

**Export to jEdit** - This item exports the current model to the jEdit text editor. Refer to [Section 3.2.1, “General Properties”](#) for more information about configuring jEdit for use by the ModelEditor.

- **Window** - The **Window** menu includes an item for the batch command interpreter window and a list of the currently open ModelEditor dialogs, such as Property Views.

## 2.1.3. The Navigator

The Navigator provides a logical hierarchical representation of the model's components and views. The root of this tree is the Plug-in node. Any model that is currently open will appear underneath the appropriate plug-in node. For example, in [Figure 2.3, "Example Navigator View"](#), note the node labeled w4Loop.med appears under the node labeled TRACE models. This indicates that w4Loop is a TRACE model.

The "name" of each model appears in parentheses to the right of its node. This is used as an abbreviated description of what the model represents. A model's name is separate from its file name to distinguish between different working versions of the same model. It can be changed by selecting the *Model Options* node underneath the model node and changing the *Name* property. The default value for the model name is "unnamed".

Each model is broken down into categories of components. The majority of these categories are Plug-in specific, however some are shared between all plug-ins. The Navigator provides access to all of the model's components including non-visual elements such as global Model Options.

Right-click pop-up menus located on the individual nodes of the Navigator can be used to add, delete or edit components, as well as to perform operations on the model.

The **Show ASCII** menu item (in the right-click pop-up menu of a View's node) will open an ASCII View showing what the corresponding model's ASCII deck would look like (only for the selected component). This non-editable view is handy for those analysts who are familiar with ASCII decks and want to verify that they are creating viable ASCII input for their codes.

**Note** The number of components of that contained within a category or sub-category is displayed in brackets following the node name. Notice in [Figure 2.3, "Example Navigator View"](#) that there are 33 hydraulic components, and two of those components are pumps.

**Note** A model's name will appear surrounded by a pair of stars if that model has changes that need to be saved. Notice that in [Figure 2.3, "Example Navigator View"](#) that the model 'unsaved' (named as such specifically for this example) has unsaved changes while W4Loop.med does not.

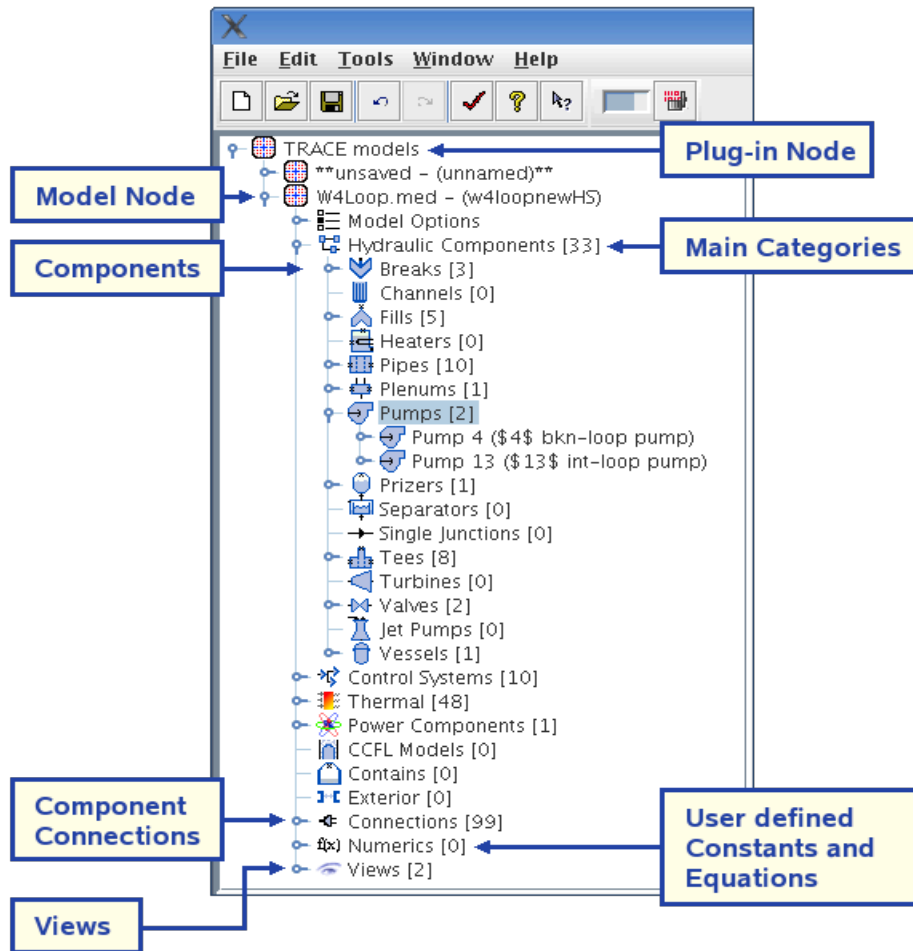




Figure 2.3. Example Navigator View

## 2.1.4. The Main Property View



The Main Property View, shown in Figure 2.4, “Example Property View”, provides the central point for viewing and editing properties of model components, Display Beans, etc. It displays the properties of the current selection from either the Navigator or a 2D View. Changes to these properties will immediately be reflected in all other open views (2D, ASCII, Property, etc.). An example of a Property View is shown in Figure 2.4, “Example Property View”.

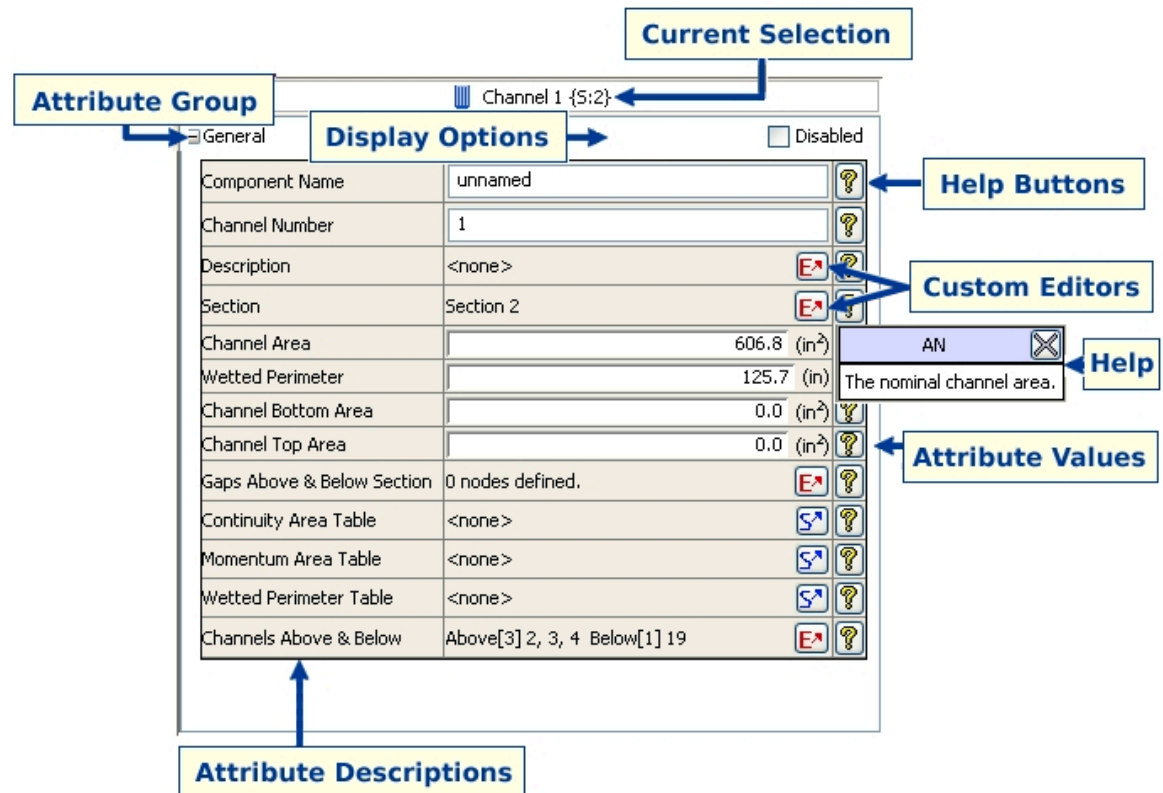
**Property Views** are made up of a set of Attribute Groups and a turn-key button  to show and hide each. Next to each button is the name of the group.

**Attribute Groups** are used to organize the properties of a component within the Property View. A turn-key button located before the Attribute Group Name is used to expand or collapse the list of attributes. Each group has a 2 column table of the attributes (description and value). While some objects have only the *General* group, others may contain several more groups.

**Help Button** - A help button  appears next to each property. Pressing this button will display a more detailed description of the property being edited.

**Disabled Check Box** - This check box (shown in [Figure 2.4, “Example Property View”](#)) is used to activate the display of disabled properties in each of the attribute groups.

Custom Editor  and Component Selection  buttons are located adjacent to some attribute values. These buttons open detailed custom dialogs for editing components and for selecting other model components where appropriate.



**Figure 2.4. Example Property View**

**Note** A separate Property View can be opened for individual components by double-clicking on the component in either the Navigator or a 2D View. The same separate Property View can also be opened by selecting the **Properties** item from the right-click pop-up menu of the component in either location.

## 2.1.5. 2D Views

2D views are a convenient way to visualize a model. Model components can be laid out on the view in a logical fashion and connections between components can be created and modified in an easy graphical manner. Views also support annotation features including text and simple graphic shapes allowing the user to build an image that represents the model of interest in a clear and visually appealing fashion. For more detailed information on annotations see [Section 2.3.3, “Creating Annotations”](#).

A 2D View can be embedded in other views to allow the user to drill-down into more complex parts of the model shown in other views. Once added to another view these embedded views will be represented by the *Display Icon* specified in the view properties. The **Add To View** menu in the 2D View's right-click pop-up menu is used embed a 2D View into another 2D View.

### 2.1.5.1. View Toolbars

The view toolbars located above the 2D View include the main view toolbar, a tools toolbar, and optional plug-in specific toolbars used to select components to be inserted into the view. The main view toolbar contains buttons for cut, copy, paste, paste special, find, group and ungroup operations. Paste special can be used to paste multiple copies of a copied set (Refer to [Figure 2.5, “2D View Toolbars”](#)). The **Lock View** button is used to lock the current view. Locking a view prevents components from being moved, copied, pasted, resized or deleted.



*Figure 2.5. 2D View Toolbars*

The tools toolbar contains buttons used for manipulating the View in various ways. These include:

#### Select Tool

The select tool is used to select, move and manipulate elements of a View. Components may be selected using the left mouse button. Using the left mouse button with the CONTROL key pressed will add or remove components from the current selection. Components may also be selected by drawing a rubber band box around the desired selection by pressing and holding the left mouse button on an empty space near the component and dragging the mouse to extend the box completely around the component.

#### Pan Tool

The pan tool is used to change the visible portion of a zoomed-in View. This feature can be used to maneuver around a large model by pressing the left mouse button and dragging. The displayed portion of the model will move with the mouse.

#### Zoom Tool

The zoom tool changes the position and scale of the View. Clicking in the View will zoom in a set amount; holding the shift-key and clicking will zoom out the same fixed amount. Clicking and dragging to select a region (drawing a box) will zoom in to the selected region. Right-clicking in the View with the zoom tool will show a menu for selecting a specific zoom position or fitting the entire View to the window.

#### Connect Tool

This tool is used to create connections between two components by selecting connection points on each component. To connect two components, place the pointer over a *from* connection point



and click the left mouse button, then move the pointer to the desired *to* connection point on another component and left click again. The mouse pointer will turn into a blue dot when a viable *to* connection point is passed over. A line representing the connection will then be drawn in the view.

## Insert Tool

The insert tool includes a button to activate the tool and a drop down menu to select the type of Annotation or component to insert. Once a type has been selected, left-clicking anywhere in the view will create an object of the selected type. After the insertion is complete the insertion tool will be deactivated.

To insert multiple components of the same type hold the CONTROL key during the insertion click. This will keep the insertion tool active and allow multiple inserts.

### 2.1.5.2. View Menu Items

View menu items can be accessed via the right-click pop-up menu of the view and via the menu bar for undocked views. The actual pop-up menu displayed will depend on the item type (and quantity) currently selected in the view. The specific code plug-ins dictate what items the right-click menus contain, however certain menu items appear frequently and will be described here. Many of the menu items available from the right-click menu in the 2D view perform identical functions to those in the Navigator.

- **Properties** - This item opens a separate property view for the selected object. This separate property view is functionally equivalent to the main property view shown in [Figure 2.4](#), “Example Property View”.
- **Show ASCII** - The **Show ASCII** menu item will open a view showing what the corresponding model's ASCII deck would look like for the selected component. This view is handy for those analysts who are familiar with ASCII decks and want to verify that they are creating viable ASCII input for their codes.

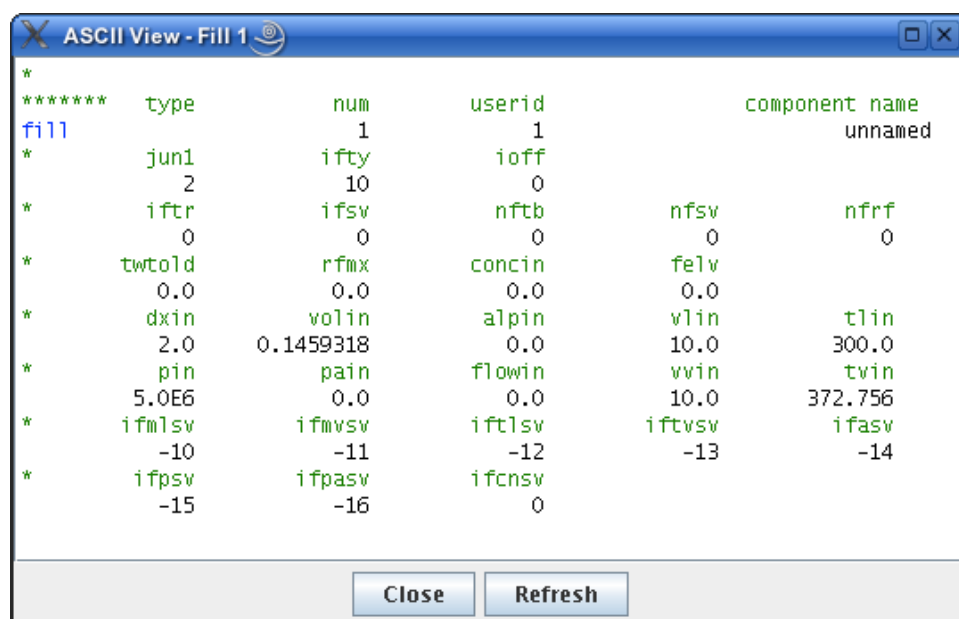








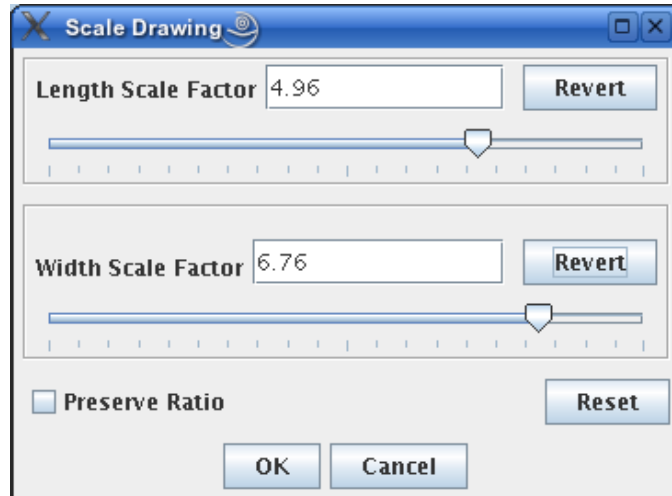
Figure 2.6. ASCII View

- **Reference Docs** - The **Reference Docs** menu item will bring up a window displaying the relevant portion of the analysis code user's guide.
- **Copy** - Copies the selected objects to the clipboard. Whenever a new cut or copy operation is performed the copy/paste buffer's contents is overwritten with the new cut/copy object and the old object is lost. (The Control-C keyboard shortcut is supported on most platforms. )
- **Cut** - Copies the selected objects to the clipboard and then removes these objects from the view. This process **does not remove components from the model** but merely removes the graphical representation of those components from the view. (The Control-X keyboard shortcut is supported on most platforms. )
- **Paste** - Pastes the graphical items (not the model components) previously copied to the clipboard. (The Control-V keyboard shortcut is supported on most platforms. )

**Note** A component can only be represented in a view by a single drawn component.

- **Paste Special** - Opens the Paste Special dialog to allow duplicates of the copied **model components** to be pasted. This dialog is a plug-in specific feature, however, most current plug-ins allow pasting multiple copies and renumbering the components as they are pasted. (The Shift-Control-V keyboard shortcut is supported on most platforms. )
- **Group / Ungroup** - The grouping feature of 2D views allows the user to create a "visual group" for a set of selected visual objects (annotations, display beans, etc.) Once created, a visual group is treated as a single object in the view. Selecting any part of the group will select the entire group. Moving, cutting, copying, etc. all affect the entire group. Groups can be dissolved at any time by pressing the Ungroup button.
- **Delete** - This item removes the currently selected item(s) from the model. A confirmation dialog is presented before the actual deletion occurs. The deleted objects are NOT saved in the clipboard. (The Delete keyboard shortcut is supported on most platforms. )
- **Align** - The Align sub-menu contains items can be used to line up objects in a view by the objects' top , bottom , left  or right  faces as well as by their horizontal  or vertical  centers.
- **To Front / To Back** - These items move the selected object either to the front or the back of the objects in the view. **To Front** ensures that no other objects will appear on top of the selected objects. **To Back** ensures that the selected objects will not appear on top of other objects.
- **Scale Drawing** - This item allows the user to change the default scaling of a drawn component. Drawn components may be scaled by length and width. Adjusting the sliders in the Scale Drawing dialog will cause the on screen display of the object to update to reflect the new scale. The user may also choose to simply enter the scaling factors by hand in the available text boxes. This scaling only affects the visual display of the component, the underlying geometry data is unchanged.

**Note** In some cases it may be more convenient to use the 2D View scaling described in [Section 2.3.4, "Component Display Scaling"](#).



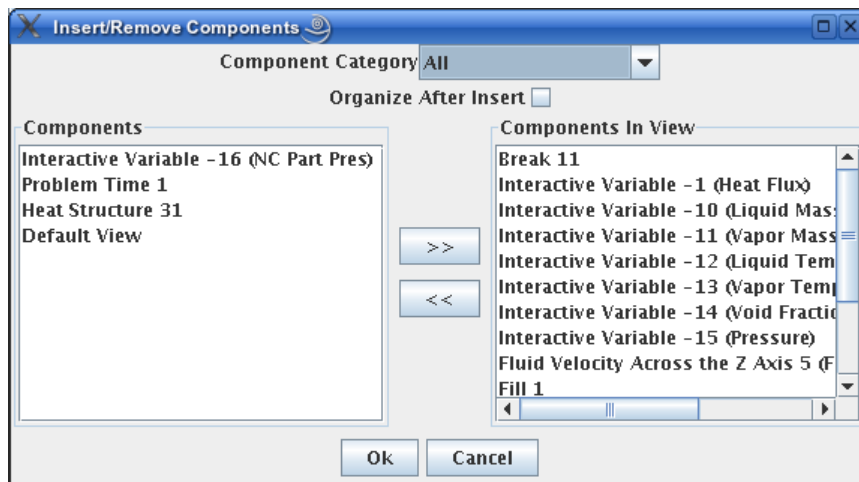
*Figure 2.7. Scale Drawing Dialog*

- **Organize** - If more then one object is selected, the **Organize** menu item may be used. This item will apply the code plug-in's layout algorithm to try to arrange the selected items in a visually clear way. Drawing organization is a plug-in specific operation that may not be supported by all code plug-ins.
- **Redraw** - Sometimes the graphical views' display is not automatically updated when a change to the model is made in another view. This item will cause the current view to be redrawn to reflect the current state of the model and its components.
- **Print View Menu** - The items in this menu allow the view to be printed. The **Entire View** item will print the entirety of the view, regardless of the current zoom scale or pan location. The **Current Perspective** item will print the only the visible portion of the view using the current zoom scale and pan location.
- **Export Image Menu** - The items in this menu are used to export the view as an image. The following formats are currently supported: JPEG, PDF, PNG, SVG, TIFF. Items are provided to export either the **Entire View** or the **Current Perspective**, similar to the Print View Menu above.
- **Select Menu** - The select menu can be used to select a sub-set of the objects in a view by type. Items are provided in the sub-menus for selecting every type that can be added to the view.
- **Zoom Menu** - This menu includes items for setting the current zoom to either **Fit To View** or the following preset scales: **%10**, **%25**, **%50**, **%75**, **%150**, **%200**.
- **Undock View** - Undocking a view removes it the ModelEditor main window and opens it in a separate window. This feature can be especially usefull for presentations as the undocked view can then be maximized to fill the entire screen. Closing and re-opening the view will return it to the docked state.
- **Close** - Closes the current view. The view can be re-opened by selecting the **Open** item in the right-click pop-up menu of the view's node in the Navigator.

### 2.1.5.3. View Tools Menu

The Tools menu contains a set of useful operations that are less often used. Elements may also be added to the tools menu by code plug-ins. This menu is available as a sub-menu of the right-click menu of the 2D View. The following are some of the more frequently available menu items in the **Tools** menu.

- **Add / Remove Components** - This item will open a dialog that allows the user to choose from a list of components in the model (show in the left hand column) to add to the view. Also, the user may select components already in the current view that they would like to remove from the view.



*Figure 2.8. Add / Remove Components Dialog*

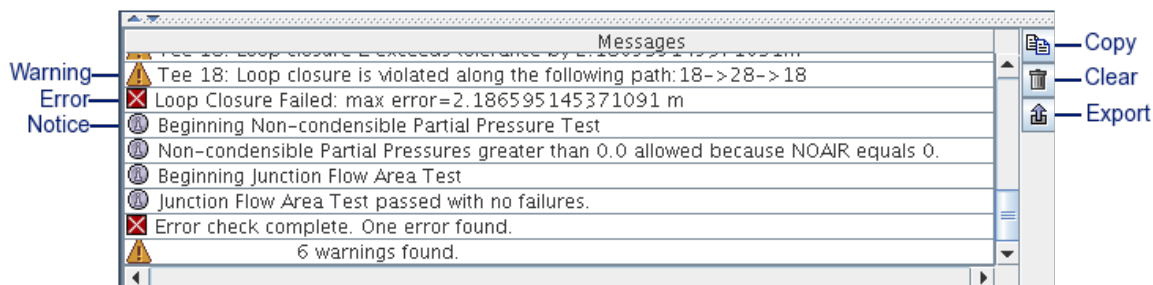
- **Show All Connections** - When components are added to a view, drawn connections are created to represent connections between the components. The drawn connections can be **Cut** from the display to simplify the view. **Show All Connections** will restore any drawn connections missing from the view.
- **Reset Connections** - This will cause all of the selected components connections to be *reset*. *Reset* means to remake the visible connection by dropping any user added points or adding points per the default connection creation algorithm.
- **Import / Export View Template** - These items allow view templates to be imported and exported for a view.

A View Template is a file that contains the annotations, component locations, etc. required to reproduce a view inside of another similar model. This allows detailed 2D views to be duplicated easily between models. This can also be used to preserve complex views for a model that must be modified outside of the ModelEditor and re-imported.

- **Trim Excess Canvas** - This feature removes the unused canvas space inside a view. This allows a view to be set to a very large size by the during the initial creation and layout and trimmed down to the minimum required size when complete.
- **Export to jEdit** - This item exports the current model to the jEdit text editor. Refer to [Section 3.2.1, “General Properties”](#) for more information about configuring jEdit for use by the ModelEditor.

## 2.1.6. The Message Window

The Message Window displays a running list of error, warning, alert and notice messages. Processes such as saving a file or checking a model will produce messages in this window. Buttons located along the right side of this window are used to clear the window, export selected messages to a file or copy selected messages to the clipboard.



*Figure 2.9. Message Window*

## 2.2. User Preferences

The SNAP ModelEditor user preferences are available from the main Edit menu. These preferences are presented as a set of General Preferences (such as Handle Size) used by all plug-ins and a set of preferences for each plug-in.

The General Preferences are divided into three Attribute Groups: General, Colors, and Window Arrangement. These are described in more detail below.

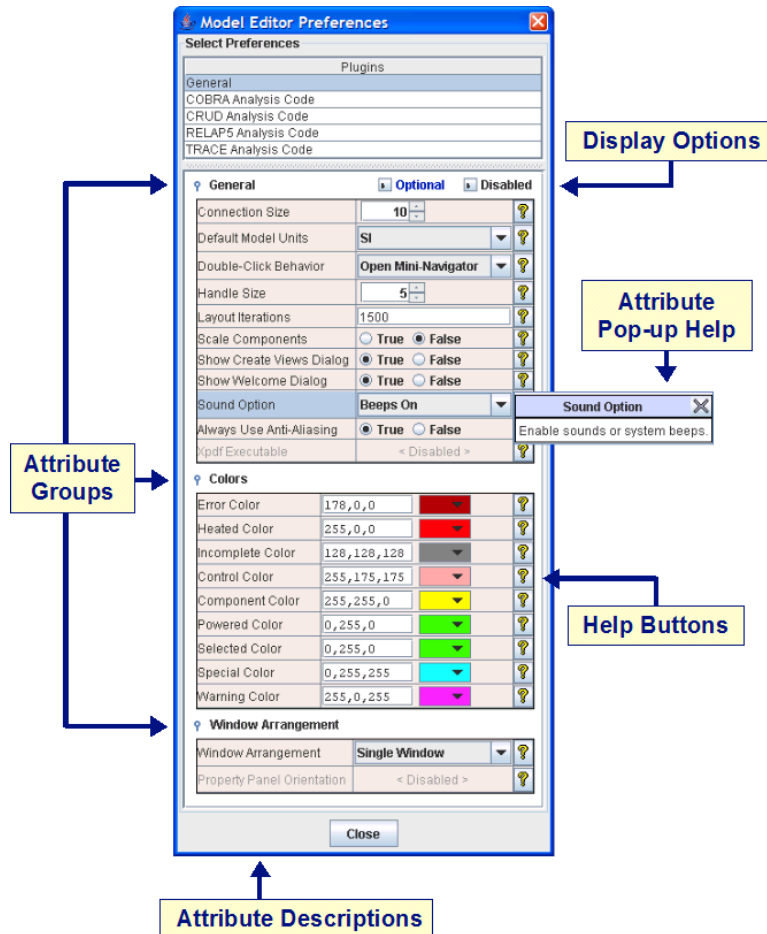
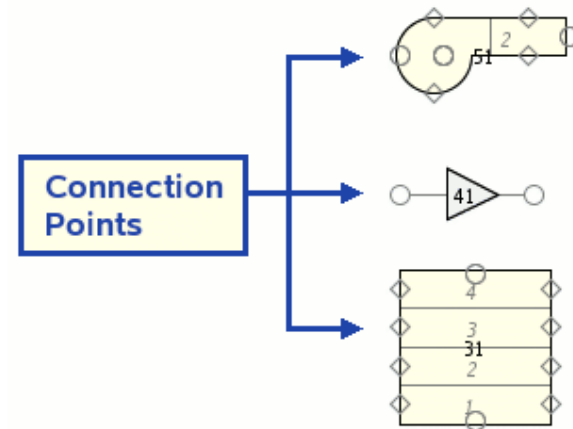


Figure 2.10. ModelEditor Preferences Dialog

## 2.2.1. General Preferences

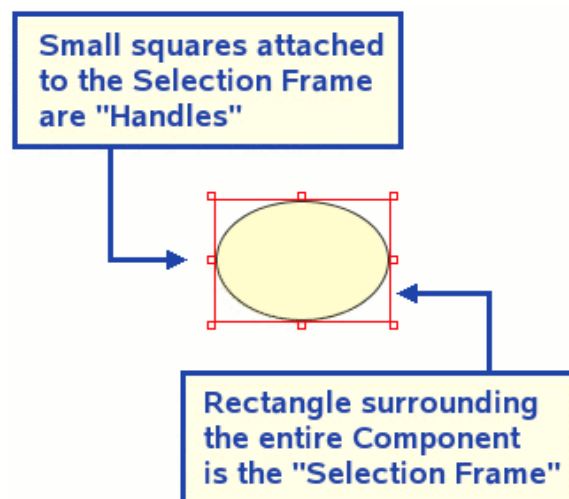
Preferences in the General category refer to a number of miscellaneous preferences, some of which shall be applied to all appropriate plug-ins.

- **Connection Size** - This value sets the size of the connection points, which is defaulted to 10 pixels. Connection points are graphic shapes providing hints to users as to where a connection to a component can be made. Different shapes may indicate how a connection can be made. For instance, in the TRACE plug-in, a circle is used to indicate a point that may be used to start a connection. A diamond is used to indicate a point that may be used to end a connection. Refer to [Figure 2.11, “Connection points used in the TRACE plug-in”](#).



*Figure 2.11. Connection points used in the TRACE plug-in*

- **Default Model Units** - Selects whether models default to using SI or British units.
- **Double Click Behavior** - When a user double clicks an object in a view, the ModelEditor will either select the object in the Navigator view or open a mini-navigator view. This option controls which behavior will occur.
- **Handle Size** - This value sets the size of “handles,” or the boxes that are drawn in the selection frame. It is defaulted to 5. See [Figure 2.12, “Handles and Selection Frame”](#).



*Figure 2.12. Handles and Selection Frame*

- **Layout Iterations** - A user may choose to “organize” a model (or selection of the model) in a graphical view. In short, organize attempts to place components on the canvas in a logical and “clean” fashion. The iteration number to be entered here determines how much time and effort should be spent on this process. Generally, higher numbers produce better layouts, but cause the organization process to take longer. The default value is 1500.

**Note** The Layout Iterations preference is used as a hint for the layout systems included with each plug-in. Thus, the effect of this preference varies from plug-in to plug-in.

- **Scale Components** - When this option is selected, components shown in a graphical view are scaled in size based on the data entered for that component (volume, length, etc.). Because small components can become difficult to see with much larger components, the user may choose to turn off scaling when components are of drastically different size.

**Note** This preference is used as a hint for plug-in 2D View rendering and is not supported by all plug-ins.

- **Show Create Views Dialog** - When set to true, a Create Views dialog will be displayed for each model imported into the ModelEditor.
- **Show Welcome Dialog** - This option controls the appearance of the welcome dialog at the ModelEditor startup.
- **Always Use Anti-Aliasing** - When set to true, all drawings in 2D Views will be anti-aliased (some drawings are always anti-aliased). Anti-aliasing greatly improves the appearance of most drawings by smoothing the curves and rough edges. Anti-aliasing incurs a performance cost, so turning this off may result in better 2D View scrolling and repaint performance.
- **Xpdf Executable** - Sets the location of the xpdf executable used to display PDFs on UNIX systems. This option is not available on Windows installations of SNAP.

## 2.2.2. Color

This section lists the different colors assigned to various components inside of a view. There are two ways to modify a color preference. The first is to click on the box displaying the color as it would appear and select a new color from the drop down menu. Second, a user can click on the text field to the left of the color box, which displays the color as three integers under the RGB color scheme. Changing these numbers and pressing Enter will change the color to the corresponding color in the RGB scheme.

## 2.2.3. Window Arrangement

This option can be set to single or multiple. When set to multiple window arrangement, the message window, component navigator and all other inner frames of the ModelEditor become undocked and independent of each other. Switching back to single arrangement recombines the views into one main window.

## 2.3. Creating a Model

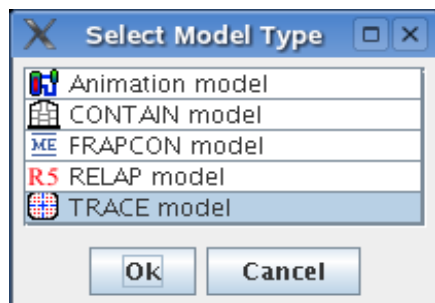
Facilitating the creation of models is the primary purpose of the ModelEditor. Models are classified in one of two categories: pre and post processing. A pre-processing model is generally comprised of components used to form the basis of a calculation which shall be submitted to a Calculation Server. Alternatively, a post-processing model is used to display the either the results of a calculation or some other form of data; in SNAP, this usually takes the form of an Animation. This section shall focus on functionality common to pre-processing models. Refer to [Chapter 7, The Animation Plug-in](#) for more information about post-processing in SNAP.

### 2.3.1. Opening a New Model

Pressing the New button on the main toolbar will open the model type selection dialog as shown in [Figure 2.13, “Example New Model Dialog”](#). This dialog allows the selection of any of the



currently loaded plug-ins for a new model. After selecting the desired type a new model will be created and added to the Navigator. A single empty view will also be created and opened.



**Figure 2.13. Example New Model Dialog**

After creating a new model special attention should be paid to its Model Options node in the Navigator as it contains several important model specific properties (refer to [Figure 2.14, “Model Options”](#)).

Editing buttons are provided to enter specified model data. Note that pressing the **E** to open a particular working window when it is already open will bring the window to the foreground.

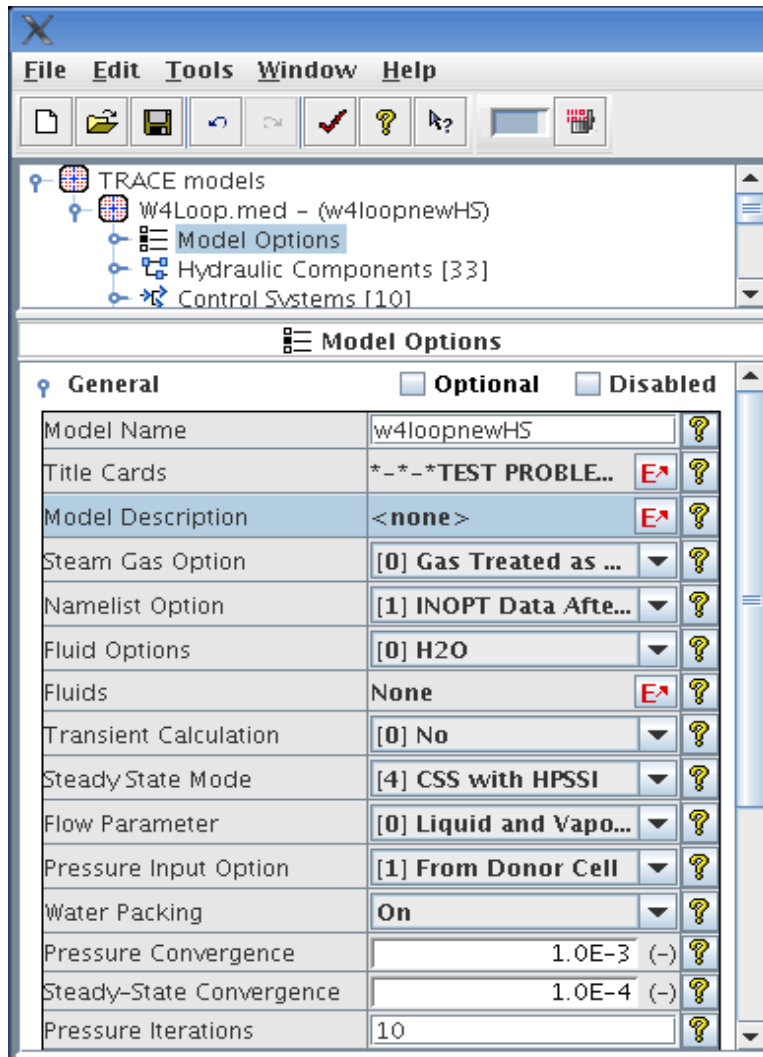
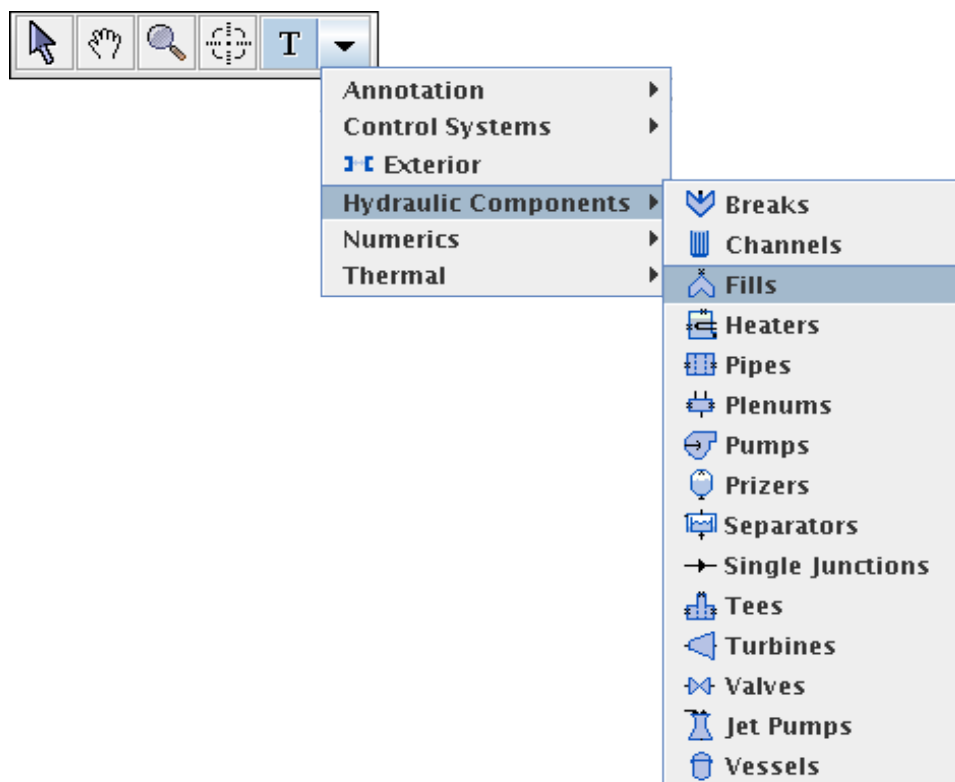


Figure 2.14. Model Options

## 2.3.2. Creating and Editing Components

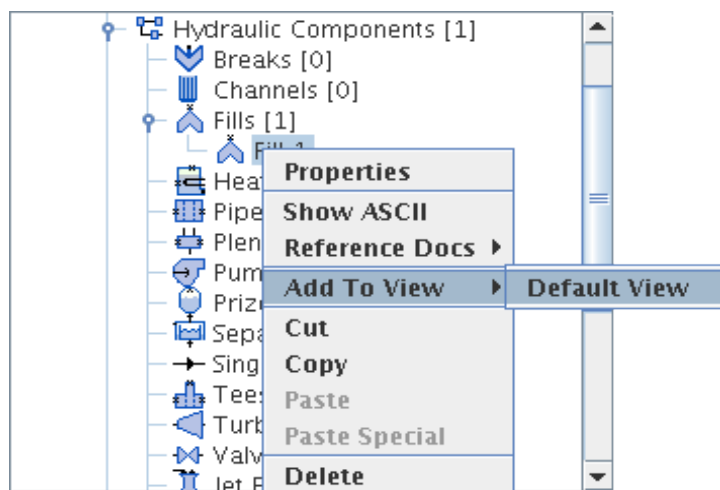
SNAP provides several methods of creating and editing components. The most common method is to use the component insertion button located above the 2D View (see [Section 2.1.5.1, “View Toolbars”](#)). This button opens a pop-up menu that contains all of the component types organized by category as shown in [Figure 2.15, “Adding TRACE Components from the View Toolbar”](#). As an example, when editing TRACE models, Pipes are located in the Hydraulic Components category, while User Constants are located under Numerics.

Once the appropriate component type has been selected the button to the left of the drop down menu button changes to the icon of that component type and the view cursor changes into a crosshair. Left-click within the 2D View to insert the component. This may open a completion dialog for the component before it is inserted into the View. The Select Tool will be automatically activated after the insertion is complete to allow the newly created component to be moved or resized. Holding the control key when left-clicking to insert a component will leave the Insert Tool active, allowing multiple components to be created without reactivating the Insert Tool for each.



**Figure 2.15. Adding TRACE Components from the View Toolbar**

The second notable method of inserting a component involves the Navigator, as illustrated for a Fill component within a TRACE model in [Figure 2.16, “Adding TRACE Components from the Navigator”](#). First, expand the component category of the desired component; in the example, this is Hydraulic Components. Next, right click on the desired component type to display a popup menu and select **New**. This creates a new component of the specified type with a default name.



**Figure 2.16. Adding TRACE Components from the Navigator**

To add the newly created component to a 2D View, right-click on the component in the listing under the component type to display a pop-up menu. From this pop-up, select the desired View to which the component is added from the **Add to View** submenu. The component will be placed within the selected View.

To edit a component, select the component, either in the 2D View or in the Navigator. The properties for that component will then appear in the Main Property View. See [Section 2.1.4, “The Main Property View”](#) for more information on editing properties.

### 2.3.3. Creating Annotations

Annotations can be used to mark-up a view with static information about the contents of that view. For example: A rectangular annotation with titled border can be used behind a group of components to visually separate them from the surrounding components. Annotations are created in a view using the Insert Tool. All Annotations appear in the Insert Tool drop-down menu under the **Annotations** sub-menu.

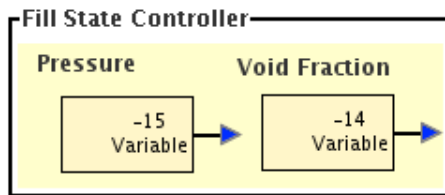


Figure 2.17. Annotations Example

The **Views** item also appears in the Annotation menu for convenience. This item creates an embedded view icon that can be used to *drill-down* to the view it represents.

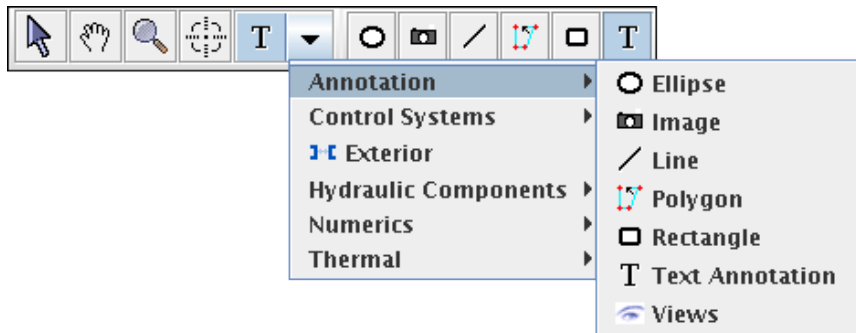


Figure 2.18. Adding Annotations from the View Toolbar

The following types of annotations are included with the ModelEditor and available for use in any view. More information about the specific properties of each annotation type can be found in the pop-up help buttons to the right of each property in the Main Properties View.

## Ellipse

The ellipse (or Elliptical Annotation) is a very simple rounded shape with an optional filled center. When inserting an ellipse, *left-clicking* will insert an ellipse of the default size. To insert an ellipse of a specific size, *left-click and drag* to create a rubber-band box that is approximately the desired size and release the mouse button.

The following properties can be specified for an Elliptical Annotation:

- *Height* - The vertical height of the annotation (in pixels).
- *Width* - The horizontal width of the annotation (in pixels).

- *Fill Background* - When this property is set to *True*, the ellipse will paint its center using the *Background Color* specified. When this property is set to *False* the center of the ellipse will be transparent.
- *Background Color* - The color used to fill the center of the ellipse when *Fill Background* is *True*.
- *Outline Color* - The color used to paint the outline of the ellipse.
- *Line Thickness* - The thickness of the line drawn around the outline of the ellipse.

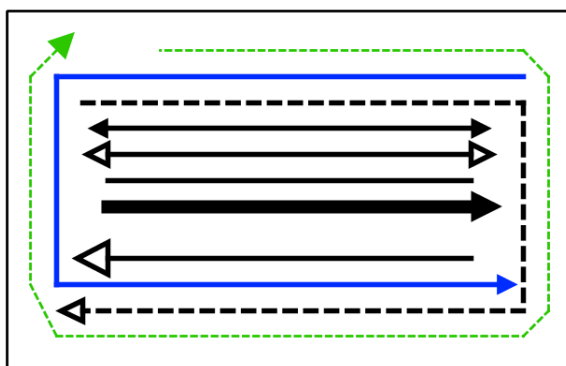
## Image

The image annotation allows an image to be displayed inside a view and it is inserted with a simple left-click. Newly created image annotations will display the default light bulb image. The following properties can be specified for an Image Annotation.

- *Height* - The vertical height of the annotation (in pixels).
- *Width* - The horizontal width of the annotation (in pixels).
- *Border* - An optional border can be specified for an image annotation. Etched, line and beveled borders are available using the provided border editor.
- *Image* - The *Image* property is used to select the image to display. Once selected, this image data will be loaded directly into the annotation and stored with the view.
- *Tooltip Text* - The image annotation includes optional tooltip text. This text is displayed when hovering the mouse over the image. Some simple HTML tags may be specified for this property but must be surrounded by `<html></html>` tags.

## Line

Line annotations are lines with optional filled or hollow arrow heads and dashing. The line thickness and relative arrowhead size can be specified as well as the line color.



**Figure 2.19. Line Annotation Examples**

Inserting a line annotation is a process of defining the line segments that will make up the line. The *first left-click* will begin the line. Moving the mouse will show that a rubber-band line is drawn to the previous point in the line. *Left-clicking again* at another location will create a line segment and begin drawing the rubber band line to that location. Additional left clicks will create

more line segments. *Double-clicking* with the left mouse button will complete the last segment and insert the line annotation using the defined segments.

At any point during the insertion process the right mouse button can be used to remove the previously added point. If there is only a single point defined then this will cancel the line insertion.

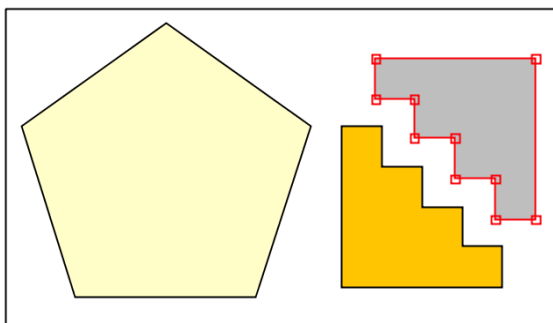
Once inserted, the points that make up a line may be relocated by left-clicking and dragging the red rectangular handle around the point. In addition, the right-click pop-up menu of the line includes the **Add Point** and **Remove Point** menu items. The **Add Point** item can be used to add a new relocatable point anywhere on the line. The **Remove Point** item can be used to remove all but the first and last points of a line. Note that the **Remove Point** item is only available when right-clicking on (or very near) a point.

- *Arrow Size* - The size of the annotation's arrow heads relative to the specified line thickness.
- *Dashed* - When this property is *True* the line will be drawn with a dashed pattern.
- *Color* - The color used to draw the line.
- *First / Second Arrow Head* - The style used to draw the line's arrow heads. Each head style can be specified individually. Available styles include None (no head), Filled (a filled triangle) and Hollow (an unfilled triangle).
- *Thickness* - The thickness (in pixels) of the line. This value also affects the size of the arrows on either end of the line.
- *Tooltip Text* - The line annotation includes optional tooltip text. This text is displayed when hovering the mouse over the annotation. Some simple HTML tags may be specified for this property but must be surrounded by `<html></html>` tags.

## Polygon

The polygon annotation is a 2D closed figure made up of any number of line segments.

Inserting a polygon annotation is a process of defining the line segments that will make up the polygon. The *first left-click* will begin the polygon. Moving the mouse will show that a rubber-band line is drawn to the previously defined point. *Left-clicking again* at another location will create a line segment and begin drawing the rubber band line to that location. Additional left clicks will create more line segments. *Double-clicking* with the left mouse button will complete the last segment, close the shape, and insert the polygon annotation using the defined segments.



**Figure 2.20. Polygon Annotation Examples**

At any point during the insertion process the right mouse button can be used to remove the previously added point. If there is only a single point defined then this will cancel the polygon insertion.

Once inserted, the points that make up a polygon may be relocated by left-clicking and dragging the red rectangular handle around the point. The vertical and horizontal line segments in a polygon may also be relocated by clicking left-clicking and dragging the line.

The right-click pop-up menu of a polygon includes the following additional items that can be used to customize its shape:

- **Flip Horizontal / Flip Vertical** - These features are used to horizontally or vertically mirror the points about the center of a polygon.
- **Reshape Polygon** - This feature is used to completely reshape a polygon into a regular polygon with a user specified number of sides and outer radius. The number of sides and outer radius is requested by an input dialog.
- **Rotate Polygon** - This rotates the polygon points clockwise a user specified number of degrees. The number of degrees is requested by an input dialog.
- **Scale Polygon** - This feature scales the points of the polygon by a user specified fraction. The scale fraction is requested by an input dialog.

The following properties can be specified for a Polygon after insertion:

- *Foreground Color* - The color used to paint the outline of the annotation.
- *Fill Polygon* - When this property is set to *True*, the annotation will paint its center the *Polygon Color* specified. When this property is set to *False* the center of the annotation will be transparent.
- *Polygon Color* - The color used to fill the background of the annotation when *Fill Polygon* is *True*.
- *Fill Background* - When this property is set to *True*, the annotation will paint its rectangular bounds in the *Background Color* specified. When this property is set to *False* the background of the annotation will be transparent.
- *Background Color* - The color used to fill the rectangular bounds of the annotation when *Fill Background* is *True*.
- *Tooltip Text* - The polygon annotation includes optional tooltip text. This text is displayed when hovering the mouse over the annotation. Some simple HTML tags may be specified for this property but must be surrounded by `<html></html>` tags.

## Rectangle

The rectangle annotation creates a rectangle with an optional filled background and border. By default the rectangle has squared corners but can be rounded by setting the *Rounded Corners* property to *True*.

When inserting a rectangle, left clicking will insert a rectangle of the default size. To insert a rectangle of a specific size, left click and drag to create a rubber-band box that is approximately the desired size and release the mouse button.

The following properties can be specified for a rectangular annotation:

- *Height* - The vertical height of the annotation (in pixels).
- *Width* - The horizontal width of the annotation (in pixels).
- *Rounded Corners* - When true the corners of this rectangular annotation will be rounded using the Rounded Arc Height and Width.
- *Rounded Arc Height / Width* - The height and width (in pixels) of the arc used to round the corners of the annotation when *Rounded Corners* is true.
- *Fill Background* - When this property is set to *True*, the annotation will paint its center the *Background Color* specified. When this property is set to *False* the center of the annotation will be transparent.
- *Background Color* - The color used to fill the center of the annotation when *Fill Background* is True.
- *Outline Color* - The color used to paint the outline of the annotation.
- *Line Thickness* - The thickness of the line drawn around the outline of the annotation.
- *Border* - An optional border can be specified for a rectangular annotation. Etched, line and beveled borders are available using the provided border editor.

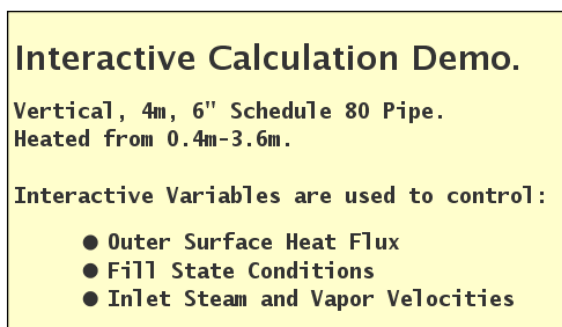
## Text

The text annotation is used to label or describe portions of a view and can be inserted with a simple left-click. Like other annotations, the text annotation can optionally fill its background with a specified color and has an optional border. The following is a list of the properties of a text annotation:

- *Tooltip Text* - The text annotation includes optional tooltip text. This text is displayed when hovering the mouse over the annotation. Some simple HTML tags may be specified for this property but must be surrounded by `<html></html>` tags.
- *Foreground Color* - The color used to paint the text entered for this annotation. Note that HTML tags specified in the *Text* of this annotation may override this color value.
- *Fill Background* - When this property is set to *True*, the annotation will paint its background the *Background Color* specified. When this property is set to *False* the center of the annotation will be transparent.
- *Background Color* - The color used to fill the background of the annotation when *Fill Background* is True.
- *Border* - An optional border can be specified for a text annotation. Etched, line and beveled borders are available using the provided border editor.



- *Text Margin* - The amount of space used to surround the text for this Text Annotation.
- *Text* - The text displayed in the text annotation. [Figure 2.21, “Text Annotation Example”](#) is an example of a text annotation using the optional HTML tag support.



**Figure 2.21. Text Annotation Example**

HTML tags can be entered directly into the text of a text annotation. These tags will be interpreted by the built-in Java HTML renderer. This allows tables, bulleted lists, font colors, font sizes, etc. to be specified directly in a text annotation. The above example can be seen by entering the following text:

```
<H2>Interactive Calculation Demo.</H2>
<CODE>
  Vertical, 4m, 6" Schedule 80 Pipe.<P>
  Heated from 0.4m-3.6m.<P><P>
  Interactive Variables are used to control:
  <UL>
    <LI>Outer Surface Heat Flux</LI>
    <LI>Fill State Conditions</LI>
    <LI>Inlet Steam and Vapor Velocities</LI>
  </UL>
</CODE>
```

## 2.3.4. Component Display Scaling

The scaling of components within a 2D View is controlled by the *Pixels Per Meter* property of the View (to display the View properties within the Main Property View, switch to the Selection Tool and select any point in the view not occupied by a component). The initial default value of 20 pixels per meter is designed to display complete model diagrams for large, full plant models. If the components appear too small to identify key details, the relative size of the components may be increased by adjusting the *Pixels Per Meter* property.

Furthermore, if increasing the scale of the View does not allow certain features of components to be distinguished, such as the cells of a pipe, it is possible to increase the relative width to height scaling for components placed in the view. Locate the View's *Width Scale Factor* property and adjust it to make components appear wider or thinner as necessary.

**Note** The 2D drawing of components can also be scaled individually using the Scale Drawing menu item as described in [Section 2.1.5.2, “View Menu Items”](#).

## 2.3.5. Checking a Model for Errors

Once a model has been completed, it can be checked for errors. To do so, select either the Check Model button from the main toolbar, or the Check Model menu item located under the Tools

menu. If the model contains no errors, the Message Window will display the message **"Error check complete. No errors found."**

If errors are discovered, the problem component can be immediately selected by double clicking on the error message inside the Message Window.

Model validation tests such as the elevation checker may be enabled and disabled by editing the Model Validation property of the Model Options navigator node.

## 2.3.6. Saving a Model

Save the model using the **File -> Save** menu item, or by utilizing the CTRL+S shortcut. If the model has not yet been named, a Save dialog will be displayed within which the name and location of the model may be selected.

**Note** ModelEditor models have a default extension of .med.

## 2.4. User Defined Numerics

The SNAP User Defined Numerics (UDN) feature is designed to allow properties of a model to be modified and/or calculated outside the normal input for the model. To this end, User Defined Constants, Variables and Functions are provided. The use and capability of each is detailed in the following sections.

The general use of numerics:

- A user defined constant is created to hold the value of particular properties. (e.g. a cell's "diameter" and "length" )
- A user defined function is created to calculate additional values. (e.g. a cell's "area" and "volume")
- A user defined variable is created to hold the result of the calculation.
- The appropriate model properties are set to the user defined constants and variables defined.

With the above usage, any change to the user defined constants would cause the user defined functions to be executed to update the user defined variables, thus updating the model properties that were set to use the constants and variables.

To create a new UDN, first expand the Navigator node *Numerics* for the current model. Then, right-click on the *User Constants*, *User Variables* or *User Functions* node to open the pop-up menu for that Category. Select the **New** item from the pop-up menu to create a new numeric of that type.

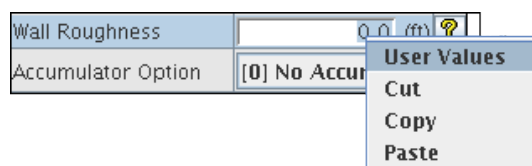
### 2.4.1. Constants and Variables

User Defined Constants (UDC) are symbolic names that may be used in place of entering floating point values for model parameters. These constants are useful for holding values that will be

changed for parametric runs. A Parametric export means that variables that are identified as parametric will have their values changed depending on a user defined table of values for each new deck exported.

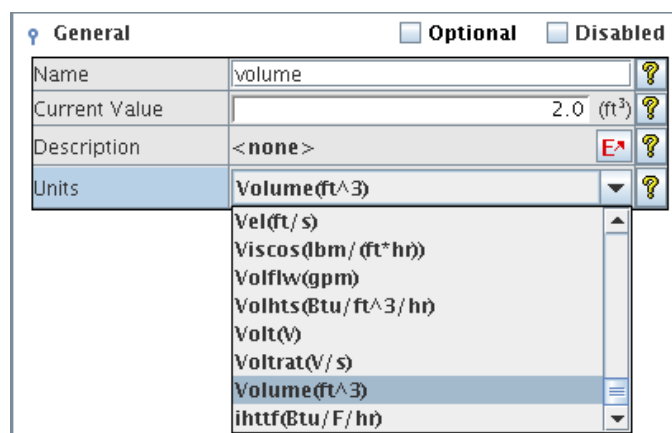
By defining model parameters using user defined constants or variables, the analyst may update common model parameters from a single location. By creating a parametric constant containing a set on input values, an analyst can efficiently generate a series of parametric analyses.

To assign a property to a user defined constant or variable simply right-click in the editor for that value and select the User Values item from the pop-up menu. A dialog will be displayed to allow the desired value or constant to be selected.



**Figure 2.22. Assigning a User Defined Numeric**

The names of user defined values are restricted only by the fact that they must not be valid numbers. For example: "-1.3" is a valid number and therefore cannot be a user defined numeric value. Names may have spaces and punctuation marks. When choosing a user defined constant or variable in an editor the choices are listed alphabetically, so wise choice of names can save time and avoid confusion.



**Figure 2.23. User Defined Variable Properties**

User defined constants can be assigned a unit type (such as length, temperature, etc.) from the units defined for by the plug-in.

User Defined Variables (UDV) are very similar to constants with the exception that they cannot be used for parametric runs and the actual value of a UDV is assigned by a user defined function. ( See [Section 2.4.3, "User Defined Functions"](#) below. )

## 2.4.2. Numerics in a View

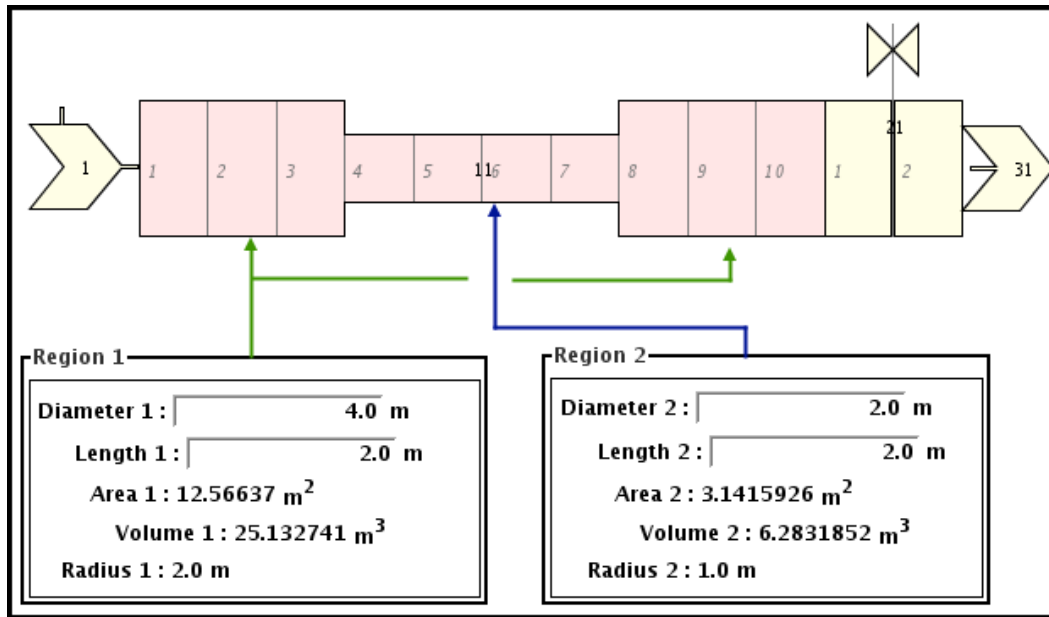


Figure 2.24. User Defined Numerics Example

User defined constants and variables can be added to a 2D view in the same way as other components: the **Add To View** menu in the numeric's Navigator node pop-up menu, the **Create View** item from the Numerics Navigator node, etc. These constants and variables are represented in a view as a *Drawn Numeric*. *Drawn Numerics* appear as a text label that displays the name of the numeric, current value and units. (e.g. "Area 1" in Figure 2.24, "User Defined Numerics Example" above.)

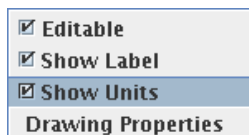


Figure 2.25. Drawn Numeric Menu

The appearance and capability of a drawn numeric can be changed by using the following items in its right-click pop-up menu. As shown in Figure 2.25, "Drawn Numeric Menu".

- **Editable** - This checkbox item enables the editing of a user defined constant by entering a new value into the Drawn Numeric. (e.g. "Diameter 1" in Figure 2.24, "User Defined Numerics Example" above.) Entering a value into a Drawn Numeric will change the value in the same way as editing the constant's properties directly.
- **Show Label** - This checkbox item enables and disables the display of the user numeric name in the Drawn Numeric.
- **Show Units** - This checkbox item enables and disables the display of the units defined for a user numeric. This option only affects the Drawn Numeric. The user numeric units will not be affected nor will Drawn Numerics in other views that refer to the same numeric.

- **Drawing Properties** - The drawing properties item opens the Drawn Numeric Properties dialog. This dialog can be used to modify the label text (replacing the name of the numeric), font, foreground & background colors and border of the drawn numeric.

## 2.4.3. User Defined Functions

A user defined function is, in essence, a Python function that is executed: when a model is opened, when a user defined constant is changed and when the **Execute Functions** item is selected. This function retrieves data from user defined constants and/or external sources and computes values that are then placed in user defined variables. User defined functions can use user defined variables and user defined constants through the following special access methods:

- **GetConstant** - The syntax for retrieving a constants value inside a user defined function is as follows:

```
X = GetConstant("Symbolic Name")
```

Where X is a simply a script variable for temporary use inside the user defined function and "Symbolic Name" is the name of the user defined constant.

- **GetVariable** - The syntax for retrieving a variables value inside a user defined function is as follows:

```
X = GetVariable("Symbolic Name")
```

where: X is a simply a script variable for temporary use inside the user defined function and "Symbolic Name" is the name of the user defined variable.

- **SetVariable** - The syntax for setting a variables value inside a user defined function is as follows:

```
SetVariable("Symbolic Name", X)
```

where: X is a simply a script variable for temporary use inside the user defined function and "Symbolic Name" is the name of the user defined variable.

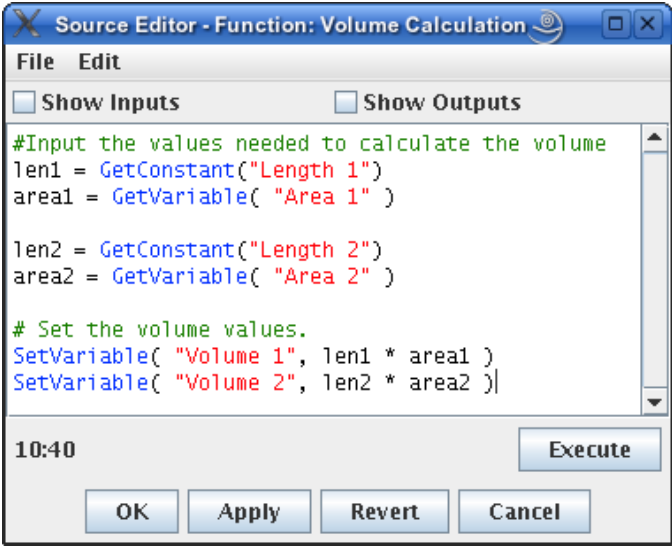


Figure 2.26. UDF Source Editor

The Python source that makes up a user defined function is edited via the *Function Source Code* property. This editor includes proper syntax highlighting for Python source as well as highlighting for the special access methods listed above.

## 2.5. Command Line Usage

The SNAP installation includes platform specific launchers for each of the included applications. For Windows a set of windows executables are included. For UNIX type systems (including OS/X) shell scripts are included. These launchers are placed in the `bin/` directory of the SNAP installation and can be executed directly from the command line. This allows the ModelEditor to be executed from other applications or scripts. When combined with batch commands ([Section 2.6, “Batch Command Syntax”](#)) this allows the ModelEditor to be used by automated systems such as analysis code test suites.

When run from the command line, certain options can be specified for the ModelEditor. The command line options can be displayed by running the ModelEditor from the command line with the option `--usage` as shown below.

```
Symbolic Nuclear Analysis Package (SNAP) ModelEditor
Version: 0.25.2
```

```
Command line usage:
  snap [--batch filename] [--debug] [--nosplash] [--remote] [--usage] [--version]
  where --batch identifies a batch file to process.
        --debug turns on debug mode.
        --nosplash turns off the splash screen.
        --remote disables double buffering for remote X displays.
        --usage prints this message and exits.
        --version prints the version info and exists.
```

<b>--batch &lt;filename&gt;</b>	The ModelEditor supports a batch command language. When run in batch mode, a batch file must provided. Refer to <a href="#">Section 2.6, “Batch Command Syntax”</a> for a detailed explanation of the ModelEditor batch command syntax.
---------------------------------	---

<b>--debug</b>	This option enables various debugging features of the ModelEditor and is recommended that this parameter only be used by plug-in developers.
<b>--nosplash</b>	This option disables the splash window display on startup.
<b>--remote</b>	The default JAVA double-buffering scheme causes a significant UI performance loss on some systems when displaying the ModelEditor from a remote machine. This option disables the double-buffering and may improve performance in these circumstances.
<b>--usage / --help</b>	This option simply displays the usage information shown above and exits.
<b>--version</b>	This options simply displays the current ModelEditor version (0.25.2 above) and exits.

## 2.6. Batch Command Syntax

The following is the standard batch command set. Each command is listed with optional fields shown in between less than < and greater than > signs, while required fields are shown between square brackets [ ].

- **SHOW** - The show command is used to display the primary user interface. This allows for batch commands to execute in such a way that the user receives immediate visual feedback. If this option is not used, the ModelEditor will run without a graphical interface.
- **HIDE** - This command hides the primary user interface.
- **MESSAGE [message]** - Where *message* contains a quoted string to be displayed to the user.

The MESSAGE command can be used to display a single quoted message to the user in a popup dialog with an OK button. This batch command will stop the batch file process until the user presses OK.

- **CHECK\_MODEL** - This performs a model check in an identical manner to the Check Model button on the main toolbar. This is useful for a batch verification of a large number of models.
- **EXIT** - The exit command is used to close the ModelEditor. This should be used at the end of a batch file to ensure that the ModelEditor process does not continue running. If the batch file contains the show command, this will close the user interface as well.
- **OPEN <Mn> [filename]** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

The open command is used to open a MED (\*.med) file in the ModelEditor. Upon opening, the new model becomes the current model. Optionally, the user may declare that the model inside that file should be identified with one of the ten labels. Acceptable values for the label field are M0 through M9. Until that model is closed, it may be accessed using its label in following batch commands. The last parameter is the file name and should include the full path to the file.

- **SAVE <Mn> [filename]** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

The save command is used to save a model in MED format. Optionally, the user may specify the model to be saved by that model's predefined label. Acceptable values for the label field are M0 through M9. If no label is given, the current model will be saved. The last parameter is the file name and should include the full path to the file.

- **CLOSE <Mn>** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

The close command is used to close a currently open model. Optionally, the user may specify the model to be closed by that model's predefined label. Acceptable values for the label field are M0 through M9. If no label is given, the current model will be closed. Any unsaved changes to the given model will be lost when closed.

- **SETCONST <Mn> [name] [value]** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

This command is used to set the current value of a User Defined Constant to the specified value. The name of the constant must be enclosed in double-quotes if the name contains any spaces.

- **EXECUDF <Mn>** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

Executes the User Defined Functions in the specified model (if given) or the current model if none is specified.

- **SET [property] [value]** - The set command is used to set a property to a specific value.

Where [property] is defined as:

- **CURRENT** - The current model. Valid values of CURRENT are M0-M9.

**Note** At this time, only the property CURRENT (the current model) can be changed.

- **LOGFILE [filename]** - The logfile command is used to specify a file where messages are logged. The filename should include the full path to the log file.
- **MACRO <Mn> [filename]** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

Where *filename* is the file name which should include the full path to the file in quotes.

This command executes the python macro contained in the specified file.

- **LOCK\_VIEWS <Mn>** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.

This command will lock all of the views for the specified model. The current models views will be locked if the model label is not specified.

- **UNLOCK\_VIEWS <Mn>** - Where *Mn* is defined as one of the ten valid model labels numbered M0-M9.



This command will unlock all of the views for the specified model. The current models views will be unlocked if the model label is not specified.

## Plug-in Specific Commands

In addition to those commands listed above, each ModelEditor code plug-in can support plug-in specific batch commands. Plug-in specific batch commands must be prefixed with the plug-in ID of the plug-in that will interpret it. For example, a TRACE import command would look like the following:

```
TRACE IMPORT ASCII_FULL "trace_model.inp"
```

## Inline File Arguments

Some plug-in specific commands require a way to specify more complex multi-line input for some commands. An *Inline File* is intended as a way to place a larger, more strictly formatted set of data as a single value argument. An *Inline File* argument must be the last argument to a batch command.

```
PLUGIN SETDESCRIPTION <
This is an example of using an Inline File to set the
description of a model.
This description can span multiple lines and will maintain
its
spacing
and formatting.
.
```

Note that the *Inline File* begins with the < (less than) character and ends with a . (period). The < should be at the end of the line preceding the Inline File and the . should be on the line after the *Inline File* on a line by itself.

---

# Chapter 3. The Configuration Tool

The Configuration Tool is used to configure global properties for the SNAP client applications. Most client application properties changed in the Configuration Tool will not take effect until the client applications have been restarted.

This tool is also used to startup, shutdown and configure the SNAP Calculation Server. Most Calculation Server property changes take effect immediately without requiring a server restart. Refer to the description of each property below to determine which properties take effect immediately.

## 3.1. User Interface

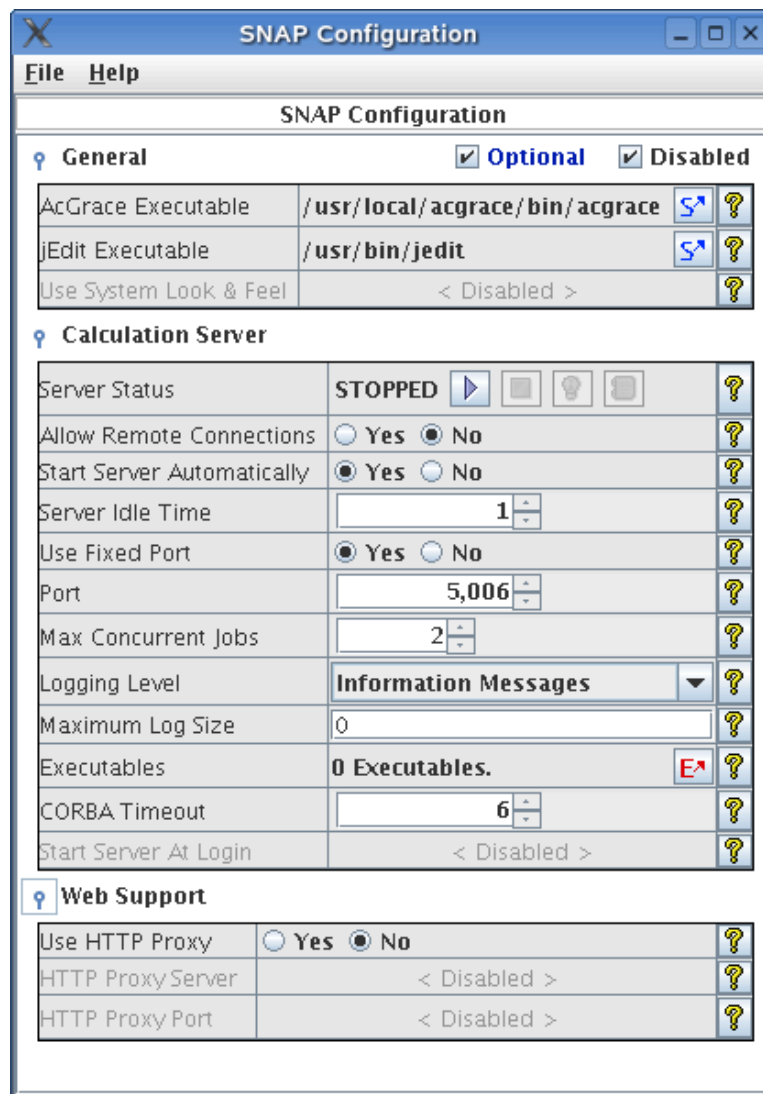


Figure 3.1. The Configuration Tool UI

### 3.1.1. Property View

The Configuration Tool uses the same property view interface used in the Main Property View of the ModelEditor. However, instead of changing parameters specific to any one component or View, properties edited in the Configuration Tool are global changes to all SNAP applications. For more information on how to use a Property View, refer to [Section 2.1.4, “The Main Property View”](#).

### 3.1.2. Menu Items

In addition to the Property View, the Configuration Tool also includes a menu bar which contains access to various functions outlined below.

- **File -> Save** - Saves the current configuration.
- **File -> Export Public Keys** - Exports the public encryption keys used for connecting to the Calculation Server from a remote location. Using this menu item opens a file selection dialog which may be used to specify the name and location of the key. Note that Outlook email program may not allow filenames that end with the .key extension.
- **File -> Generate Server Keys** - Regenerates the public encryption keys used for connecting to the Calculation Server from a remote location. All previously exported keys will be invalidated.
- **Help -> Check for Updates** - Displays a login dialog which can be used to login to the SNAP web server and check for updates and optionally retrieve and install them.

## 3.2. Configuration Tool Properties

The Configuration Tool options are separated into three attribute groups: General, Calculation Server and Web Support.

### 3.2.1. General Properties

General Properties are used for options applicable to all applications in the SNAP suite.

### Plotting Tool

The path to the plotting tool used by client applications to plot data. For example, right clicking on an animation component in the ModelEditor will display a pop-up menu with a **Plot Data** menu item. Selecting this item will launch another dialog within which Data Channels specified for that component are listed. After selecting one or more channels and pressing the **OK** button, the specified *Plotting Tool* will be launched. Data specific to the selected channels will be provided to this application for plotting. SNAP currently supports the AcGrace and APTPlot plotting packages.

**Note** For AcGrace under Windows, cygwin's bin and usr\X11R6\bin directories must be included in your system path, i.e. "c:\cygwin\bin;c:\cygwin\usr\X11R6\bin;" without quotes.

## Executable

The location of the jEdit executable used by SNAP for displaying ASCII data.

jEdit is a pure-Java, programmer's text editor that was chosen for use with the SNAP system because of its stability and multitude of features. For more information on using jEdit refer to [Chapter 8, Using the jEdit Plug-in](#). For instructions on installing jEdit or the jEdit plug-in refer to [Appendix A, Installing the jEdit Plug-in for SNAP](#).

## Use System Look & Feel

When set to *Yes*, SNAP applications will be displayed with system-specific window decorations. When set to *No*, the platform independent look and feel will be used. Turning the system look and feel on requires an application restart.

*Use System Look & Feel* is only available on platforms for which the Configuration Tool is able to detect a system-specific look and feel.

## 3.2.2. Calculation Server Properties

Calculation Server properties are used to configure the Calculation Server used to run analysis codes and retrieve data from completed calculations. In addition, the Calculation Server can be started from the Configuration Tool.

### Server Status

The Server Status property can be used to start or stop the Calculation Server, as well as obtain information about an active server. It is displayed in [Figure 3.2, “Stopped Calculation Server”](#) and [Figure 3.3, “Started Calculation Server”](#). The buttons on both properties are, from left to right: Start Server, Stop Server, Show Server Status, and Open Log Viewer.



*Figure 3.2. Stopped Calculation Server*



*Figure 3.3. Started Calculation Server*

- **Start Server** - This button activates the Calculation Server. It is not available when a Calculation Server is running.
- **Stop Server** - This button activates the Calculation Server. It is not available unless a Calculation Server is running.
- **Show Server Status** - This button displays a dialog containing information about client applications connected to the currently active Calculation Server. It is not available unless a Calculation Server is running.
- **Open Log Viewer** - *This feature is not yet implemented.*

## Allow Remote Connections

When set to 'Yes' the Calculation Server will allow connections from other users and other machines. To allow remote connections from other users, first change *Allow Remote Connections* property to Yes and restart the calculation server. Then, export the server's public key to a convenient location by selecting the **Export Public Key** item from the **File** menu. This key can then be transferred to those users who require access to the server.

**Note** This feature is only supported under Java version 1.5 or above.

## Start Server Automatically

When activated ('Yes'), the *Start Server Automatically* preference will enable the local server to startup when needed by local client applications. When not activated ('No'), the user will be prompted to startup the server instead. Client applications are considered local only if they are running on the same computer as the server (referred to as "localhost") and running as the same user as the server.

This property will enable the local server to startup automatically:

- when submitting a job
- when browsing the local server via Job Status
- when selecting a run for an animation from the local server
- when connecting an animation to the local server

Activating the *Start Server Automatically* preference enables the *Server Idle Time* preference.

## Server Idle Time

The maximum amount of time (in minutes) that the Calculation Server will stay active while no clients are connected. Specifying 0 for this value will prevent the server from shutting down automatically.

Note that the Calculation Server must be restarted for this value to take effect.

## Use Fixed Port

When set to 'Yes', the Calculation Server will always be started listening to a specified port. Setting this value to 'No' indicates that the Calculation Server should use the first available port it finds. Setting this value to 'No' is recommended.

Changing this property does not take effect until the Calculation Server has been restarted.

## Port

The port number that the Calculation Server will listen on when Use Fixed Port is set to 'Yes'. This is also the only time that the Port parameter is available.

Changing this property does not take effect until the Calculation Server has been restarted.

## Max Concurrent Jobs

Sets the maximum number of jobs that the Calculation Server will execute simultaneously. Any other jobs submitted after this maximum has been reached will wait in a job queue for the first available opening.

## Logging Level

This parameter determines the level of detail used in writing information to the calculation log files.

## Maximum Log Size

The maximum size (in bytes) of the log files created by the calculation server. After reaching this limit, the current log file will be renamed calculation\_server\_1.log and logging will continue in a new calculation\_server.log. An unlimited log file size can be set by specifying 0 for the maximum size.

Changing this property does not take effect until the Calculation Server has been restarted.

## Executables

The executable definitions for the Calculation Server. To submit a job to the Calculation Server for a particular plug-in, at least one executable must be defined for that plug-in.

To add an executable:

1. Click the **E** (edit) button next to the Executables attribute. This will open the Edit Executables dialog and allow the executables to be added, removed and modified.
2. Click the Add button near the center of the dialog. This will create a new executable in the list in the top half of the dialog. The attributes of the newly created executable will be shown in the bottom half of the dialog.
3. Define the attributes of the new executable.
  - **Plug-in** - The plug-in (i.e. "code type") that this plug-in will be used with. Select the appropriate plug-in for this executable from the combo box to the right.
  - **Exec ID** - A short (one word) name for the executable. This name is used only for distinguishing between executables of the same analysis code type. For instance: RELAP5 version MZB might be called "Relap-MZB".
  - **Description** - A one line description of the executable.
  - **Executable** - The absolute path to the executable.
  - **Executable Arguments** - Additional command line arguments to pass to the executable.
  - **Supports Interactive Execution** - Set to true if this executable (and it's corresponding server plug-in) support the SNAP interactive command system. Executables that support

interactive execution will allow the user to send commands to the calculation to control its execution (such as pause, resume or end).

- **Default to Interactive Execution** - Set to true if jobs submitted using this executable should default to using the SNAP interactive command system. This option is only available if Supports Interactive Execution is set to 'Yes'.
- **DeMUX Executable** - The absolute path to the demultiplexer for this executable. This demultiplexer can be executed from Job Status to create a demultiplexed data file for use by plotting packages. Access to a demultiplexed file greatly speeds the process of plotting very large data sets.
- **DeMUX Arguments** - Additional command line arguments to pass to this executable's demultiplexer. Refer to the demultiplexer documentation for a listing of the available command line arguments.

## CORBA Timeout

The timeout value (in seconds) used by CORBA connections between SNAP applications. SNAP communication over dialup connection or networks with very high traffic may sometimes exceed the maximum timeout and fail. In these cases the timeout must be increased. However, increasing this value will cause some operations (such as checking for an unavailable local server) to take much longer to complete.

The default value is recommended in most circumstances.

## Server at Login

When set to 'Yes' a shortcut will be placed in the 'Startup' group of the user's 'Start Menu.' This option is only available on Windows platforms.

### 3.2.3. Web Support

Web support properties are used in configuring how SNAP attempts to look for updates and other web related tasks.

- **Use HTTP Proxy** - Set this property to 'Yes' if the Web Support Services (such as update) should use an HTTP proxy server.
- **HTTP Proxy Server** - The HTTP Proxy server to use for all Web Support communication.
- **HTTP Proxy Port** - The HTTP Proxy server port to use for all Web Support communication.



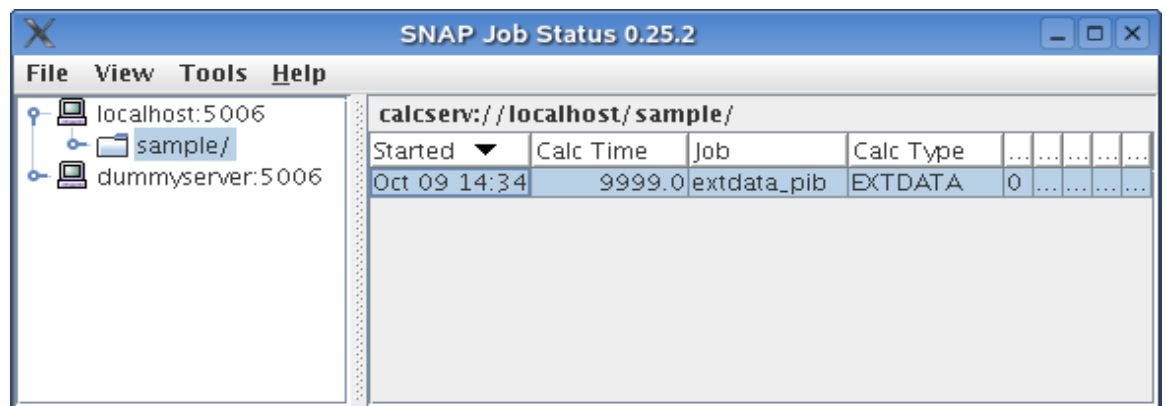
---

# Chapter 4. Job Status

The Job Status application is a client application which details the status of jobs submitted to a Calculation Server. It also provides support for adding connections to remote Calculation Servers and importing local files into the list of available jobs.

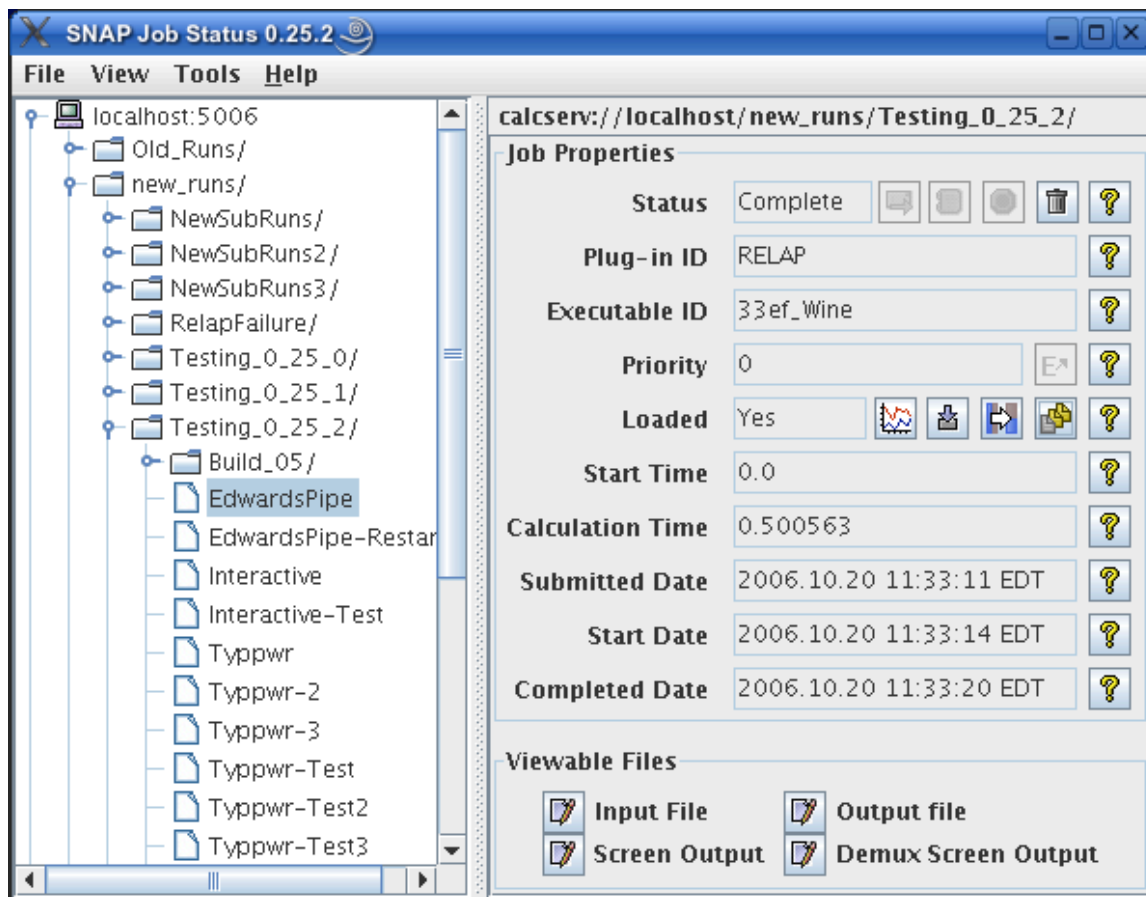
## 4.1. User Interface

The Job Status User Interface (UI) is composed of a Job Navigator on the Left and a Job Panel on the right which can display either the Job Table (shown in [Figure 4.1, “The Job Status UI”](#)) or the detailed view of the currently selected job (shown in [Figure 4.2, “Job Status Job Details”](#).)



**Figure 4.1. The Job Status UI**

The Job Navigator is a tree-style view representing a logical hierarchy of servers and root folders. Each server can be expanded into a list of mounted folders, which can in turn be expanded into a list of subfolders. Selecting an item in this view changes the contents of the Job Panel. In general, selecting a folder displays any jobs contained within that folder in the Job Table, while selecting a job displays the detailed view for that job.



**Figure 4.2. Job Status Job Details**

Right-clicking on components in the Job Navigator will display a pop-up menu. A Calculation Server's pop-up menu will contain the following menu items:

- **Connect** - Attempt to make a connection to the server. This may fail if the server is unavailable or the specified server key is invalid (see [Adding a Server](#)).
- **Disconnect** - Disconnects from the server.
- **Reconnect** - Disconnects from the server and attempts to reconnect. This option is equivalent to using the **Disconnect** and **Connect** options above.
- **Mount Root Folder** - Opens a file selection dialog for specifying a folder on the local Calculation Server which should be mounted as a root folder. For more information on this functionality, refer to [Section 4.3, "Root Folders"](#).

**Note** Root folders cannot be mounted or unmounted for remote Calculation Servers.

Right clicking on a folder in the Job Navigator will also display a pop-up menu. A folder's pop-up menu will contain the following menu items:

- **Create Folder** - Creates a new folder within the directory. Selecting this option opens a dialog which requests a name for the new directory.
- **Rename Folder** - Opens a dialog for renaming the folder. This option is not available for root directories.

- **Remove Folder** - Removes a folder from the Calculation Server. This option is not available for root and non-empty directories.
- **Import Local File** - Imports a plot file from the selected folder into the local Calculation Server. For more information, refer to [Section 4.5, “Importing a Local File”](#).
- **Unmount [folder name]/** - Unmounts the folder, removing it from the Job Navigator. This option is only available to folders that have been mounted as root folders, it is not available for subfolders.

A pop-up menu is also available for job files. However, as the menu is identical to that found in the menu bar under the Tools menu, that functionality will be described in the section, [Section 4.2, “Menu Items”](#).

## 4.2. Menu Items

The menu bar is broken into four menus, **File**, **View**, **Tools**, and **Help**.

### File

- **Sounds Check Box** - Enables and disables Job Status sounds.
- **Add Server** - Opens a dialog for adding Calculation Servers (refer to [Section 4.4, “Adding a Remote Server”](#)).
- **Remove Server** - Removes the server currently selected in the Job Navigator.
- **Exit** - Exits the application.

### View

- **Job Table Check Boxes** - Various columns in the Job Table can be hidden or shown with the check boxes found in the **View** menu, .
- **Refresh** - Refreshes the status of all components in both the Job Navigator and the Job Panel.

### Tools

The tools menu contains entries specific to the currently selected job.

- **Console Output** - Displays the Console View detailing the status of the selected job. This option is only available for calculations that are currently active (i.e. running, interactive, paused, etc.)
- **View Output** - Displays a list of viewable files for the selected job. For some jobs, this list may be empty. This option is only available when the job has been loaded.
- **Plot** - Opens the selected job file for plotting. The application used to open the file is specified by the *AcGrace Executable* property in the Configuration Tool (see [Section 3.2.2, “Calculation Server Properties”](#)). This option is only available when the job has been loaded.
- **Demultiplex** - Produces a demultiplexed copy of the selected job file. The application used to demultiplex the file is specified by the *DeMUX Executable* property of a plug-in executable (see [Plug-In DeMUX Executables](#)). This option is only available when the job has been loaded.

- **Send Command** - Displays a dialog for sending an interactive command to the selected job. This option is only available interactive calculations.
- **Terminate** - Terminates the selected job. This option is only available for jobs that are currently active.
- **Change Priority** - Changes the priority of the selected job. This priority determines the order in which jobs are started by the Calculation Server. Otherwise, jobs are selected based on the order of their submission into the queue. This option is only available for jobs waiting to be started in the job queue.
- **Load Data** - Loads a completed job into memory. Most options in the **Tools** menu can only be activated after a job has been loaded.
- **Unload Data** - Unloads a job, freeing any system resources required in keeping the job loaded.
- **Release** - Releasing a job removes the Calculation Job File (.cjf) but does not remove any of the job's data files.
- **Delete** - Removes the job from the Calculation Server. This removes any data files associated with the job.

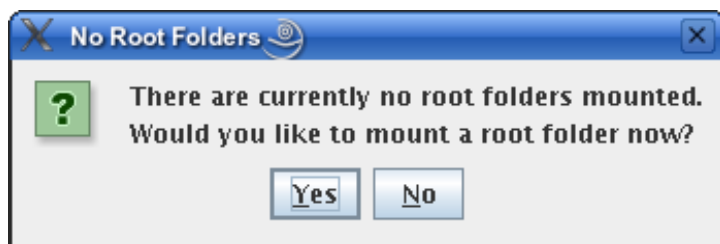
## Help

- **Help Topics** - Displays the SNAP help set section on Job Status.
- **Included Technologies** - Displays a dialog listing various technologies included in SNAP and their licenses.

## 4.3. Root Folders

The Calculation Server includes support for multiple Root Folders (with sub-folders) that can contain any number of jobs. These Root Folders can be mounted in Job Status by selecting the Mount Root Folder item from the right-click pop-up menu of the local Calculation Server node ("localhost"). These folders can be unmounted by selecting the **Unmount** item from the right-click pop-up menu of the folder. Unmounting a folder does not remove any data contained in that folder.

To mount or unmount folders, the server's node must first be expanded to make a connection to the server. If no root folders are mounted on the local server a prompt will be shown allowing the user to mount one (Figure 4.3, "No Root Folders Prompt").



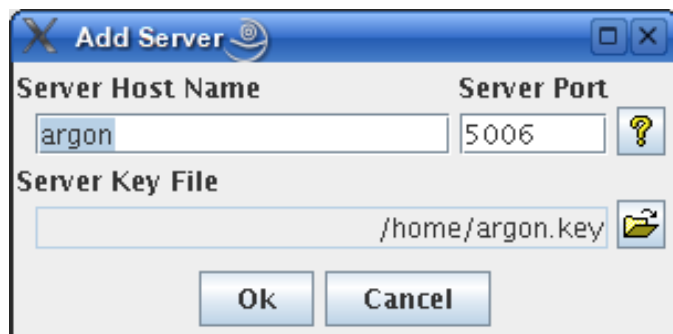
*Figure 4.3. No Root Folders Prompt*

The Calculation Server also supports any number of sub-folders under each root folder. Sub-folder can also contain any number of sub-folders, and so on.

Sub-folders can be created by selecting the parent folder and selecting the **Create Folder** item from the right-click pop-up menu. These sub-folders can also be renamed or deleted by selecting the Rename Folder or Delete Folder item from the right-click pop-up menu of the folder. Note that only empty folders can be deleted.

## 4.4. Adding a Remote Server

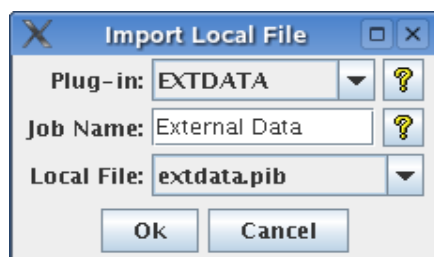
After selecting **Add Server** from the **File** menu, the Add Server dialog is displayed (see [Figure 4.4, “Add Server Dialog”](#)). To add a remote server, first enter the server's host name or IP address in the *Server Host Name* field and their calculation server port in the *Server Port* field. Then, select the Server Key File received from the remote user that started the calculation server. After pressing **OK**, the remote server will appear in the server list in Job Status and in the Animation plug-in's job selector. For more information on exporting the required Server Keys, see [Exporting Server Keys](#).



*Figure 4.4. Add Server Dialog*

## 4.5. Importing a Local File

Plot files can be imported into the local Calculation Server directly by using the Import Local File dialog available from the **Import Local File** item ([Figure 4.5, “Import Local File Dialog”](#)). To import a plot file in this manner simply select the mounted folder that contains the file then right-click and select the **Import Local File** item. The Import Local File dialog will be shown to allow the plug-in, job name and local file to be selected. Pressing **OK** will import the file into the local Calculation Server and load its data for use in animations.



*Figure 4.5. Import Local File Dialog*

---

# Chapter 5. Submitting a Job

The submit dialog (shown in [Figure 5.1, “Submit Dialog”](#)) allows the submission of jobs from jEdit or the ModelEditor to local and remote Calculation Servers. This dialog can be accessed from the ModelEditor by selecting the **Submit Job** item from the main **Tools** menu. From jEdit, this dialog can be accessed by selecting the **Plugins -> Snap -> Submit Calculation** menu item.

**Note** When submitting from jEdit, the calculation type may be selected (just below *Server*). When submitting from the ModelEditor the calculation type is determined by the model being submitted.

**Submit Calculation**

Server: localhost:5006

Executable: 33ef\_Wine

Target Folder: /Tutorial/

**Run Options**

Name: ☒ EdwardsPipe-Restart

Overwrite: ☒ Yes ☐ No ☐ Prompt

Priority: 5

Interactive: ☒ True ☐ False

Start Paused: ☐ True ☒ False

**Parametric**

☐ Enabled

First: < none >

Second: < none >

**Restart**

☐ No Restart ☐ Selected Restart File ☒ Selected Restart Job

Restart Job: /Tutorial/EdwardsPipe

☐ Submit Into Separate Folder

**Animation**

☒ Connect Animation After Submit

/home/user/EdwardsPipe\_anim.med

☒ View Console Output


Submit Cancel

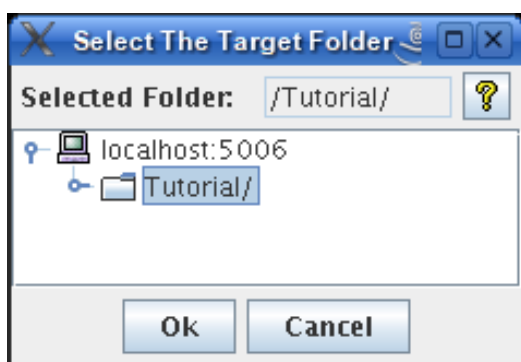
Figure 5.1. Submit Dialog

---

To submit a job, first select the target *Server* from the list of available Calculation Servers (refer to [Section 4.4, “Adding a Remote Server”](#) for a description of adding and removing servers). Selecting a server will automatically retrieve the list of available executables for the current calculation type. Select the desired executable from the *Executable* list.

**Note** When submitting to the local Calculation Server, if the server is not currently active it will be started automatically based on the current setting of *Start Server Automatically* (the section called “[Start Server Automatically](#)”).

The *Target Folder* must be selected next by pressing the  button. This will open a folder selector for the selected server. If the target server is the local calculation server then the mount and unmount folder pop-up items will be available in this folder selector. Refer to [Section 4.3, “Root Folders”](#) for a more detailed description of root folders.



**Figure 5.2. Target Folder Selector**

The next section of the dialog is *Run Options*. This section includes *Name*, *Priority*, *Interactive* and *Start Paused*. *Name* is the requested job name. If no name is specified (by un-checking the check-box to the left) then a name will be automatically generated using the plug-in name and the current date/time. If a job already exists with the requested name an overwrite confirmation will be shown to allow the existing job to be deleted before submission.

**Note** Additional run options may be included by the server code plug-in. These plug-in specific options will appear just below the *Start Paused* option. For example: the TRACE plug-in includes the *No Dif*, *No Rand* and *Run Stats* boolean run options (not shown).

The *Priority* property controls the order that jobs will be started on the Calculation Server. Higher priority jobs (larger priority) will be started before lower priority jobs.

The **Interactive** and **Start Paused** options determine whether the analysis code will be executed in *Interactive Mode*. This mode uses the SNAP runtime communication with the analysis code to send interactive commands (if supported) to the job and receive notifications of the job's current calculation time.

The **Start Paused** causes the Calculation Server to send the `&pause` message to the job as soon as it is launched. This is especially useful for animating an interactive job as it allows the animation to be connected before the job progresses.

The remaining run options are executable-specific options from the server plug-in that are used to control the analysis code execution.



**Figure 5.3. Submit Parametric**

The *Parametric* section controls the submission of a parametric set of jobs. If enabled, one or two User Defined Constants may be selected for use in generating a set of parametric runs. If the first and second constants are selected, runs will be submitted that represent every combination of the values of both variables. If only the first constant is selected, a set of runs will be submitted that represent each value of the first constant.

As an example, consider a model (EdwardsPipe-Restart) that contains 2 parametric User Defined Constants (FlowArea, and HydraulicDiam) each with the values {0.9, 1.0} (ft). These two UDCs are used to determine the flow are and hydraulic diameter of a simplified Edward's Pipe model. Submitting this model with the first constant set to FlowArea and the second constant set to HydraulicDiam would result in the following jobs.

EdwardsPipe-Restart_R01_01	FlowArea is 0.9 and HydraulicDiam is 0.9.
EdwardsPipe-Restart_R01_02	FlowArea is 0.9 and HydraulicDiam is 1.0.
EdwardsPipe-Restart_R02_01	FlowArea is 1.0 and HydraulicDiam is 0.9.
EdwardsPipe-Restart_R02_02	FlowArea is 1.0 and HydraulicDiam is 1.0.

**Figure 5.4. Submit Restart**

The *Restart* section controls whether the submitted input file will be considered a restart and from where the restart information will be retrieved. If the **Selected Restart File** option is chosen a file can be selected from the local file system to be sent to the server for the restart. If the **Selected Restart Job** option is chosen, a job can be selected from the target server to be the restart job. This option will ensure that the submitted job will not be started until the restart job completes and will copy the selected job's restart file into the appropriate location for the restart.

The *Animation* section allows the selection of an Animation mask to open after this job is submitted. After the submission the master data source of the selected mask will be changed to point to the newly submitted run and the mask will be connected.

When the **View Console Output** option is selected a Console Dialog will be opened for the submitted job. This console can be used to view the console output, pause, start, stop and terminate the job.

---

---

# Chapter 6. The Calculation Server

The Calculation Server is a remote or local application responsible for processing calculations and other data oriented tasks. A user will not typically interact directly with the Calculation Server, as it runs in the background waiting for use by a client application. Most applications in the SNAP suite are such client applications: they request services from a Calculation Server, which in turn performs the requested service, possibly sending the client data or information depending on the nature of the service. The two prevalent uses of the Calculations Server are to execute an analysis code with a provided set of input, and to retrieve data from completed calculations and external data files.

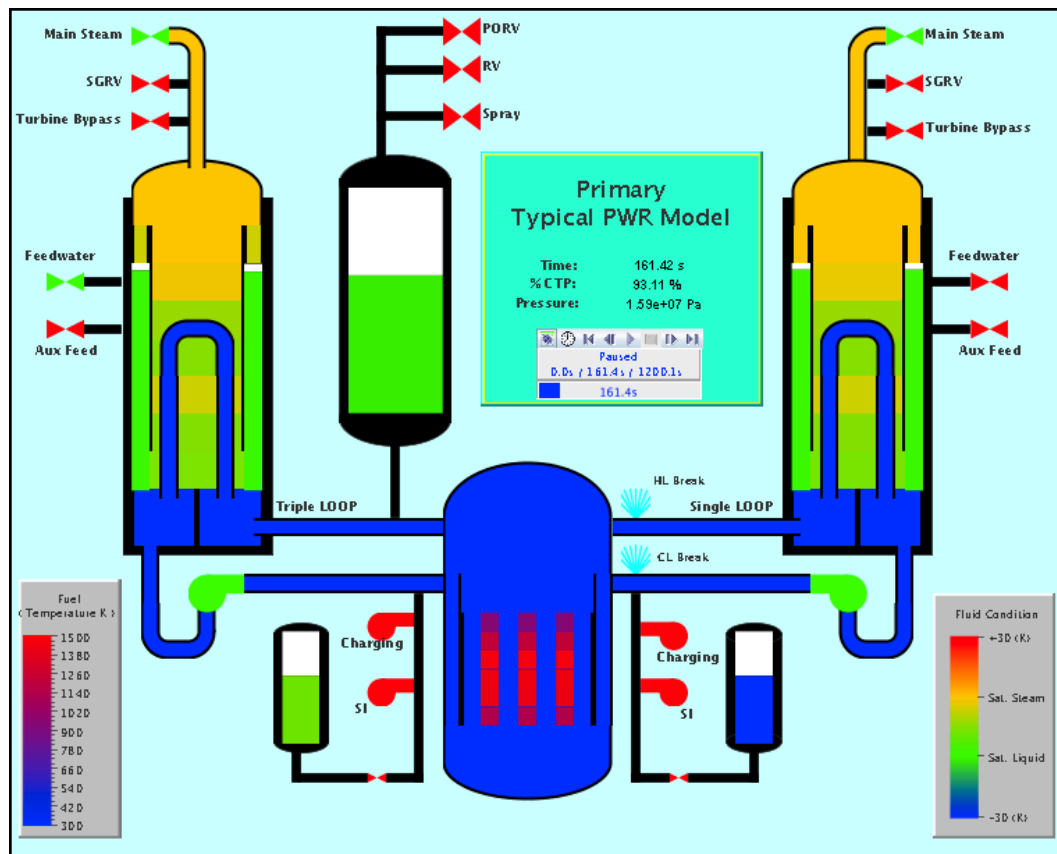
A Calculation Server is considered local if it runs as the same user and on the same machine relative to the client applications accessing it. Such a local server is can be started from the [Chapter 3, \*The Configuration Tool\*](#), but by default will start automatically when required by a client application (see the section on [Section 3.2.2, “Calculation Server Properties”](#) for more information on how the Calculation Server is started). Conversely, a Calculation Server is considered remote if the client applications requesting services of it are run on a different machine or by a different user.

For information on configuring local and remote Calculation Servers, refer to [Chapter 3, \*The Configuration Tool\*](#) and [Chapter 4, \*Job Status\*](#).

The Calculation Server makes extensive use of the Calculation Job File (CJF) format for storing information about each job. Knowledge of this format is not necessary for the normal use of the Calculation Server, however the details of this format may be invaluable to developers and test suite authors. A detailed description of this format is found in [Appendix B, \*Calculation Job Files\*](#).

---

# Chapter 7. The Animation Plug-in



*Figure 7.1. Example Animation Mask*

SNAP's interactive and post-processing capabilities are predominately realized within its animation displays. Within such a display, the results of a calculation may be animated in a variety of ways.

Animation masks are very similar to models. They are composed of Views containing a number of objects, each of which contain properties that may be edited in the Main Property View. However, instead of submitting jobs to the Calculation Server, an animation display retrieves data from the server and represents it visually in some fashion. This data can be from an actively running calculation, a completed calculation, imported EXTData files, etc.

## 7.1. Creating a New Animation Model

Pressing the New button on the main toolbar will open the model type selection dialog as shown in [Figure 7.2, "Example New Model Dialog"](#). Select the Animation Model option. Afterward, a new animation model will be created and added to the Navigator. A single empty view will also be created and opened.

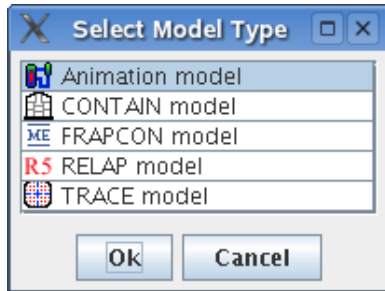


Figure 7.2. Example New Model Dialog

## 7.2. Animation Components

Animation models contain only a small number of component types: Data Sources, Ranges Numerics and Views. Creating and editing these components is identical to other component in the ModelEditor. (Refer to [Section 2.3.2, “Creating and Editing Components”](#) for more information.)

The majority of the objects in an animation model are Display Beans and Annotations within 2D views. Refer to [Section 7.8, “Display Beans”](#) and [Section 2.3.3, “Creating Annotations”](#) for more information on the creation and use of Display Beans and Annotations.

## 7.3. Data Sources

An animation requires one or more Data Sources to animate data. A Data Source is most often a reference to a job on a Calculation Server. While animating, the ModelEditor will retrieve data from the Calculation Server and display visual representations of that data. This process repeats until either the data is exhausted or the user interrupts the animation.

Each Animation Model requires one Master Data Source and may have any number of Slave Data Sources. Slave sources are other sources of data from a Calculation Server whose data will be interpolated relative to the Master Data Source.

To specify a job for a data source, first select the Data Source from the Data Sources category in the Navigator. Then locate the *Source Run URL* property in the Main Property View and press the red **E** button to launch the Select Data Source dialog (shown in [Figure 7.3, “Select Data Source Dialog”](#)).

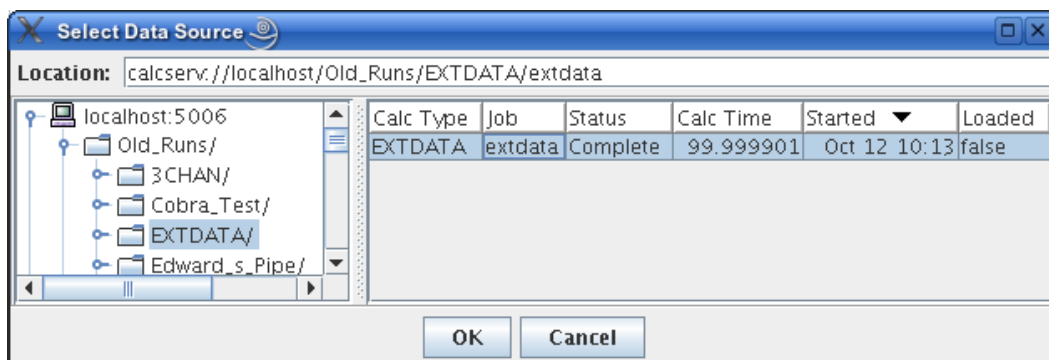


Figure 7.3. Select Data Source Dialog

Expanding each Calculation Server node in the job tree (on the left) will show the root folders for that server. Expanding each folder will show any contained sub-folders and jobs. Selecting a folder will display the list of jobs in that folder (to the right). Selecting a single job in the tree will show the details of that job to the right. Note that the *Location* field is updated when selecting jobs. This location is the URL to the job that will be used for the Data Source. For example: [Figure 7.3, “Select Data Source Dialog”](#) shows a selected job (extdata) that is an external data file managed by the EXTDATA plug-in.

**Note** Unlike other Animation components, Data Sources cannot be added to a view.

## 7.4. Sequenced Data Sources

The Animation Plug-in's Data Source component allows multiple Source Run URLs jobs to be specified as a sequence. To enable this feature, set the the *Number of Source Runs* property to the number of jobs in the sequence (up to 4) then select each job in the sequence (as described in [Section 7.3, “Data Sources”](#)) for the First, Second, Third and Fourth run URL properties.

Master (Interactive-2)	
<b>General</b> <input type="checkbox"/> Optional <input type="checkbox"/> Disabled	
Name	Master ?
Include in Animation	<input checked="" type="radio"/> True <input type="radio"/> False ?
Master Source	<input checked="" type="radio"/> True <input type="radio"/> False ?
Number of Source Runs	Two Source Runs ?
First Run URL	calcserv://localhost/Local... E ?
Second Run URL	calcserv://localhost/Local... E ?
Minimum Included Time	<input type="checkbox"/> ?
Maximum Included Time	<input type="checkbox"/> ?

**Figure 7.4. Data Source Properties**

When using a sequence of jobs the playback time will begin at the first job's start time with the first job as the current job. When the playback time reaches the start time of the next job it will become the current job and re-initialize all Display Beans to animate that job's data. The same process will occur at the start time of the third job, and so on.

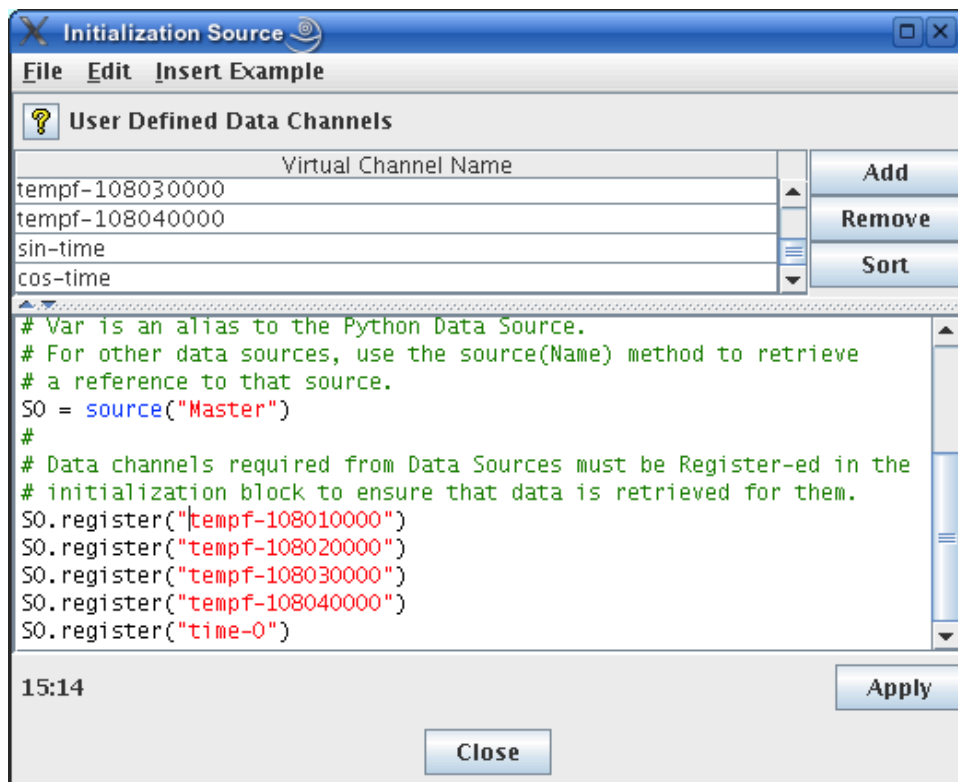
**Note** Only the last job in a sequence can be actively running. The preceeding jobs must be completed.

The channel name list for a sequence is the union of the channel lists of the jobs included in the sequence. Channels that are not available in the current job during an animation will behave in the same manner as other undefined channels. (For example: The single volume bean is colored flat gray.) When the current job changes to a job that includes the previously undefined channel it will be animated with the currently available data as normal.

## 7.5. The Python Data Source

The Python Data Source is used to animate data values that must be calculated based on data from one or more Data Sources. This Data Source contains two segments of user-defined Python code that are executed at different times.

- The *Initialization Source* is the block of Python source code executed upon activation of this Python Data Source. This code should include:
  - Registering required channels from other Data Sources.
  - Setting initial values for Virtual Data Channels in this Python Data Source.
  - Defining methods and global variables to be used in the Transient Source during animation.
- The *Transient Source* is the block of Python source code executed after each timestep of data is retrieved from the master data source. This code should include:
  - Retrieving current values of required data channels in other Data Sources.
  - Setting new values for Virtual Data Channels in this Python Data Source with `setChannel` calls.



**Figure 7.5. Python Data Source Source Editor**

Both of these source segments are edited by using the source editor shown in [Figure 7.5, “Python Data Source Source Editor”](#). The source editor provides a syntax highlighted text editor for editing python code. The list of virtual data channels available from this python data source is at the top of the source editor. Any data channels that will be made available from the python source must be added to this list.

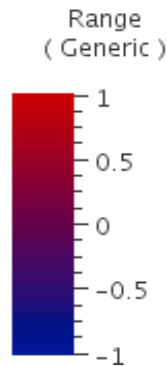
Example python code for the various data source specific methods can be found in the *Insert Example* menu. Included in this menu are: *Get a Source*, *Register a Data Channel*, *Get a Channel Value* and *Set a Channel Value*.



## 7.6. Ranges

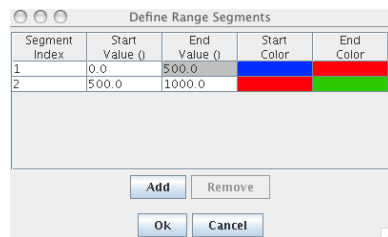
A Range is a user defined range of colors and a corresponding user defined range of values used for Display Bean animation. During an animation, a typical behavior of a Display Bean is to change color based on the current value of its associated Data Channel. For example, a Single Element Display Bean is depicted as a rectangle. If it is tied to a Generic Range with a single segment ranging between values of 0 and 1, with respective colors of red and blue, when animating a value of .75, the rectangle will deep red. Alternatively, when animating a value of .25, the rectangle will display deep blue. Any values exceeding the minimum and maximum values of the range will be displayed by the colors defined for the minimum and maximum values (alternatively, colors can be specified for out of range values).

As Ranges are themselves components, any number can be created and optionally added to a 2D View (see [Figure 7.6, “Generic Range”](#)).



**Figure 7.6. Generic Range**

The properties available for a Range are dependent on its current *Range Type*. Also, the properties available for each type are completely separate from those for other types. For example: the **Segments** specified for the *Temperature* type is independent of the **Segments** specified for the *Liquid Temperature* type. This means that *Range Type* can be used to animate different properties for a set of Display Beans. The **Segments** of a range are consecutive regions of increasing value that may have different starting and ending colors. This allows for different regions of the overall range to have separate color values. The dialog for editing a range's segments is displayed in [Figure 7.7, “Range Segments”](#).



**Figure 7.7. Range Segments**

## Temp / Pressure / Quality / Void Fraction

These range types work with beans that use either a Volume ID or a Data Channel to retrieve animation data. These types use a sets of min/max values and min/max colors to determine the displayed color for each bean. For beans that use a Volume ID, the channel that is displayed

will be determined by the specific client side plug-in managing that data source. (e.g. TRACE, RELAP5, etc.)

## Generic

This range type allows minimum/maximum values (and colors) to be specified along with SI and british units for these values.

Generic is the most flexible range type for Display Beans that use the *Volume ID* property. When these beans use a range of type Generic they will use a user-specified pattern to determine which data channel to animate. These patterns can be specified for each plug-in currently loaded (i.e. one for TRACE, another for RELAP5). Refer to the pop-up help for the *Channel Name Patterns* property for more information on the pattern syntax.

## Fluid Condition

This range type displays the current fluid conditions of a volume (by *Volume ID* only ) as a set of color ranges. Separate color ranges are specified for subcooled, saturated and superheated liquid. For this range type, the superheat and subcool values are specified as a number of degrees from the saturation temperature.

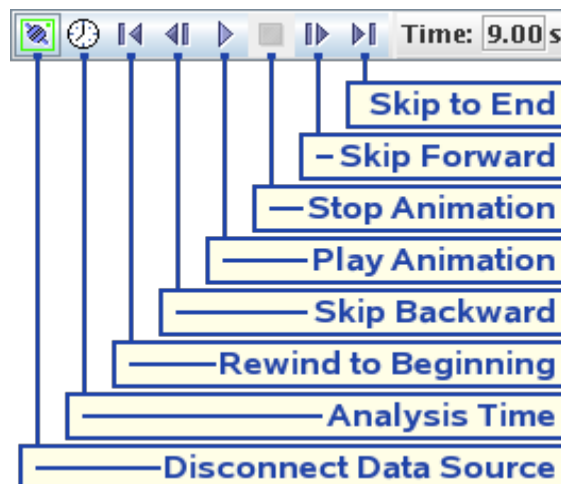
**Note** The Fluid Condition range type can only be used with beans that use a *Volume ID*.

## 7.7. Animation Playback



**Figure 7.8. Playback Controls, No Data Source Connection**

When an Animation Display is open, the Animation Playback Controls will become available in the Main Toolbar. The playback controls are a collection of buttons used to drive an animation. They are illustrated in [Figure 7.8, “Playback Controls, No Data Source Connection”](#) and [Figure 7.9, “Playback Controls, Connected to a Data Source”](#).



**Figure 7.9. Playback Controls, Connected to a Data Source**

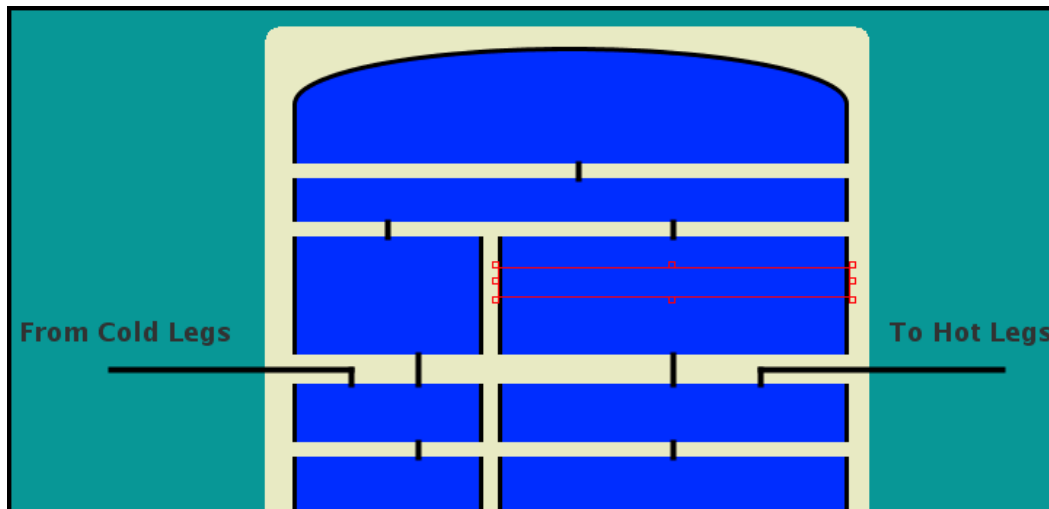
- **Connect Data Source** - Connects the Animation to the currently defined Data Sources. Once connected to a Data Source, this button becomes the Disconnect Data Source button. Only one Animation Model may be connected at a time.
- **Disconnect Data Source** - Disconnects from the current Data Sources.
- **Analysis Time** - Opens the Analysis Time dialog. This dialog is used to select an arbitrary time to seek to as well as configure various time-related settings for the animation.
- **Rewind to Beginning** - Moves the animation back to the first available time value.
- **Skip Backward** - Moves the animation backward by a fixed number of time steps. This number of time steps can be configured in the Analysis Time dialog by changing the *Skip Forward/Back Steps*.
- **Play Animation** - Begins playback of the animation. Once pressed, this button becomes a pause button which can be used to pause playback.
- **Stop Animation** - Halts the currently animating job. The stop button is only available when the job is currently active (executing) and the current playback is at the last available timestep.
- **Skip Forward** - Moves the animation forward by a fixed number of time steps. This number of time steps can be configured in the Analysis Time dialog by changing the *Skip Forward/Back Steps*.
- **Skip to End** - Moves the animation forward to the last available time value.

**Note** Skip Forward and Skip Backward are not supported by all plug-ins.

## 7.8. Display Beans

Display Beans are objects that can be added to a 2D view to display values retrieved from a job on a Calculation Server or an external data source. Display Beans are created using the Insertion Tool in the exact same way as Annotations. Display Beans can also be resized, reshaped and moved in a similar fashion to annotations.

**Note** For most plug-ins the quickest way to create a simple display is to copy the contents of a 2D View (e.g. a set of interconnected pipes or control systems) and paste into an Animation view.



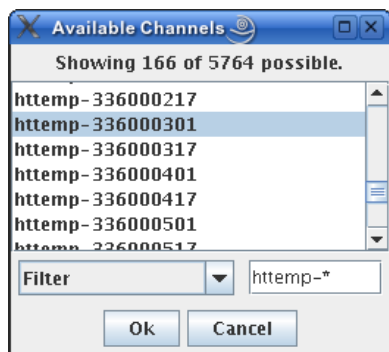
**Figure 7.10. Display Beans Example**

Each bean displays its data in a different way but they all share a common set of functionality. The following is a list of features supported by most Display Beans. In the case that a beans does not support one of these features it will be noted in that bean's description.

Single Volume			
<b>General</b>	<input type="checkbox"/> Optional	<input type="checkbox"/> Disabled	
Data Source	Master (Typypwr-2)		
Range	Fluid Condition Range		
Volume ID	350020000		
Command Menu	-not set-		
Foreground Color	0,0,0		
Orientation	Vertical		
Outline Width	3		
Volume Shape	Rectangle		
ToolTip Text	volume-350020000		

**Figure 7.11. Single Volume Properties**

- **Data Source** - All Display Beans connect to at least one Data Source, some more than one. Those beans that use more than one Data Source include pop-up explaining what aspects of the bean will use each source. Refer to [Section 7.3, “Data Sources”](#) for more detailed information on Data Sources and Data Source selection.
- **Range** - A Range is used to map a data value (or values) to a color for animation. This is useful for providing a overall impression of the progress of a calculation. Refer to [Section 7.6, “Ranges”](#) for more detailed information on Ranges and Range selection.
- **Channel Name** - The name of the data channel (or often data channels) that will be animated by the bean. Display beans that use a single data channel will most often use the channel selector shown in [Figure 7.12, “Channel Selection Dialog”](#). Beans that use multiple data channels (such as the Axial Plot bean) include more customized selectors for the bean type.

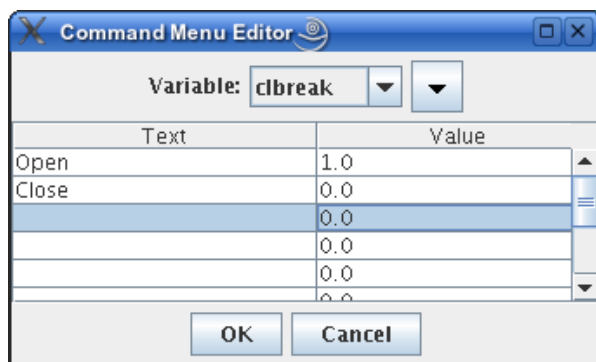


**Figure 7.12. Channel Selection Dialog**

- *Volume ID* - The Volume ID property is used by some Display Beans to to animate sets of data channels based on the current type of the Range selected for the bean. This is most often used for properties such as void fraction, liquid temperature and fluid conditions where the data channel names can be implied based on the Volume ID and range type. In these cases, the client code plug-in (for the analysis code being animated) is responsible for determining the required data channels.

**Note** Beans that use the Volume ID property do not use the Channel Name property, and vice versa.

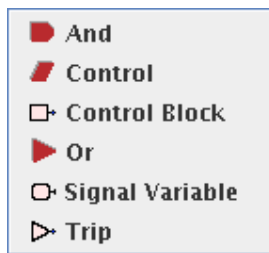
- *Tooltip Text* - This is the tooltip that is displayed when the mouse is over the bean. Optionally, this tooltip text can be a valid HTML document but if so it must include the `<html>` and `</html>` tags.
- *Command Menu* - The command menu an easy way to send interactive commands to a calculation during animation. This feature allows the user select an interactive variable and specify a list of values (and corresponding text for each name) for a Display Bean. Once defined, these values can be selected from the **Commands** sub-menu of the right-click pop-up menu of the bean. Selecting any of the menu items in the **Commands** menu will send the corresponding value to the calculation as an interactive command.



**Figure 7.13. Command Menu Editor**

The following sections describe the Display Beans included with the Animation plug-in broken down by their sub-menu location in the Insertion Tool pop-up menu. Additional beans may be installed separately or included with other pre-processor plug-ins. Refer to the individual plug-in users' manuals for more information on the beans they include.

## 7.8.1. Control System



**Figure 7.14. Control System Beans**

The control systems beans are intended for use in creating logical representations of control systems for animation. These beans do not actually perform any logic operations but merely display the value of their selected data channels.

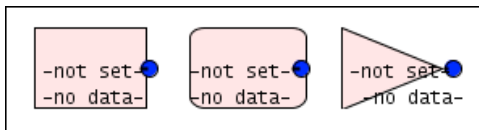
### 7.8.1.1. And / Or / Control



**Figure 7.15. And / Or / Control Beans**

The And, Or and Control beans (shown above) are more customized versions of the Single Element bean that can be used to layout rough control systems for animation.

### 7.8.1.2. Control Block / Signal Variable / Trip

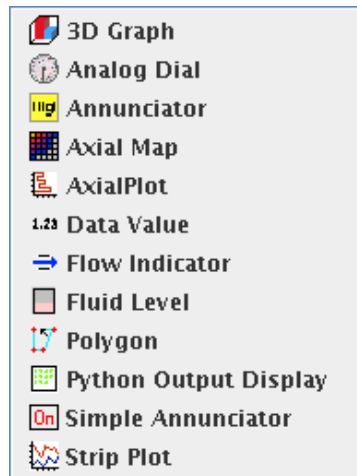


**Figure 7.16. Control Block Signal Variable and Trip Beans**

The Control Block, Signal Variable and Trip beans are intended can be used to layout logical control systems such as those in the TRACE and RELAP5 plug-ins. These beans can be created via the Insertion Tool or by pasting a control system type component into an animation view from a code plug-in that supports control system display bean creation (such as TRACE or RELAP5).

These bean support the standard single data source and data channel along with a numerical format (for the float display) and a label to replace the data channel name when animating. In addition, SI and British units strings can be provided for display depending on the units setting for the animation model.

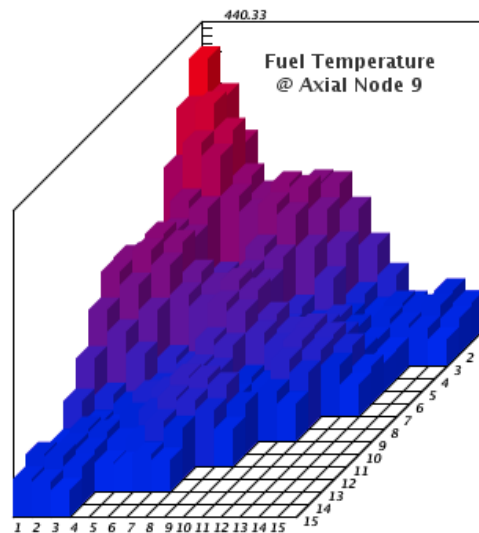
## 7.8.2. Indicators



**Figure 7.17. Indicator Beans**

The Indicators group contains a set of more specialized beans for plotting and data display.

### 7.8.2.1. 3D Graph



**Figure 7.18. 3D Graph Bean**

The 3D Graph bean is a three dimensional representation of data values similar in nature to the Stacked Element bean. Individual cells in the grid may be disabled, allowing pattern-based selection of data channels to skip unneeded cells. Orientation, grid spacing, and relative lengths of the X, Y, and Z axes can be customized. The example shown in [Figure 7.18, “3D Graph Bean”](#) displays fuel temperatures, where all empty spaces in the grid are disabled cells.

1. Insert a new 3D Graph bean using the Insert Tool.
2. Set the number of columns and rows in the bean with the *Dimensions* property.

3. Disable any unneeded cells by editing the *Enabled Cells* property. This displays a dialog of toggle buttons which can be used to set the enabled/disabled status of each cell in the grid.
4. Specify data channel names for each element in the bean.

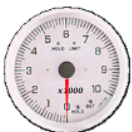
Open the *Channel Nodes* dialog by editing the *Data Channels* property of the bean. The table at the top of this dialog contains the channel names for each element in the bean layed out by row and column in the same order as they will be rendered by the bean. These channels can be input individually (by editing each table cell) or input in groups by seleting the **Use Pattern** check box and specifying a **Pattern**.

5. Select an appropriate *Range* for the bean.

In addition to the normal bean customization, the 3D Graph bean provides the following additional properties:

- *Orientation* - Determines the relative angles and lengths of the X, Y, and Z axes.
- *Grid Origin* - Determines the position of the grid along the Z axis.
- *Display Wire Mesh* - When set to *True*, a wireframe mesh is used to display data. When set to *False*, the default bar graphs are displayed.
- *Wire Thickness* - The thickness of lines used in displaying a wire mesh.
- *Column Spacing* and *Row Spacing* - Determines the relative widths of columns and rows. Editing either property opens the *Grid Spacing* dialog, which displays the current spacing values.
- *Display Grid* - When set to *True*, lines indicating the borders of cells in the grid are displayed.
- *Display Opaque Grid* - When set to *True*, the cells of the grid are filled with the color specified by the *Grid Base Color* property.
- *Display Labels* - When set to *True*, numbers indicating the index of cells in the grid are displayed along the X and Y axes. If the *Position Labels on Axes* property is set to *False*, the labels will be displayed along the bottom and right lines of the grid parallel to the X and Y axes.
- *Display Ticks* - When set to *True*, tick marks will be displayed along the Z axis. The spacing of these tick marks are determined by the specified Range.

## 7.8.2.2. Analog Dial



**Figure 7.19. Analog Dial Bean**

The Analog Dial is a 0-10 representation of a single data channel as a dial. A scale factor must be set to translate the value of the data channel into the 0-10 range.



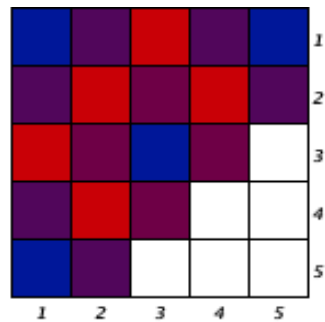
### 7.8.2.3. Annunciator



*Figure 7.20. Annunciator Bean*

The Annunciator bean is a more complete version of the Simple Annunciator that includes High, High-High, Low and Low-Low values.

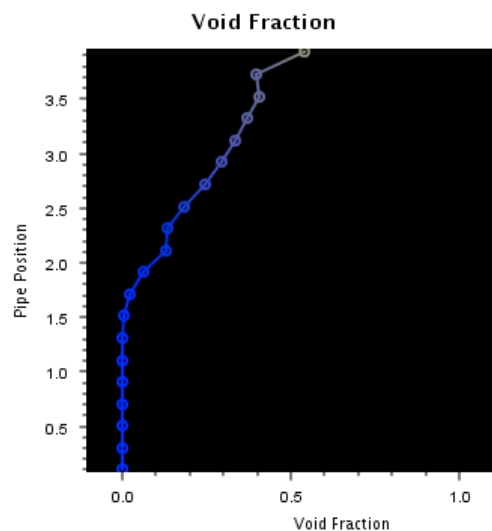
### 7.8.2.4. Axial Map



*Figure 7.21. Axial Map Bean*

The Axial Map bean is a specialized case of the 3D Graph bean. Its functionality is identical to a 3D Graph bean with a Z axis length of 0 and both X and Y angles set to 0. See the section on the 3D Graph for more information on properties specific to this bean.

### 7.8.2.5. Axial Plot



*Figure 7.22. Axial Plot Bean*

The Axial Plot bean displays a line plot based on a set of data channels for each timestep rendered. If desired, multiple data channel sets can be input to display a line plot for each set. Each data

channel set consists of a number of x/y pairs of data channels that determines the location of each point on the plotted line. Optionally, the x channel may be input as Fixed X Positions for situations where there is no x data available as data channels or the x data does not change over time.

A large number of properties are available in the Axial Plot bean to customize the plot, line and axis appearance. Refer to the pop-up help available next each property for more information on axial plot customization.

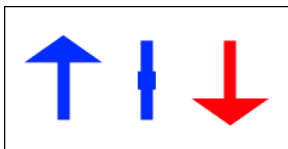
### 7.8.2.6. Data Value



**Figure 7.23. Data Value Bean**

The Data Value bean simply displays the numerical value of the selected data channel. The font, foreground color, border, etc. can be customized like most Display Beans. Also, a numerical format may be specified (in C printf format) to control how the data value is displayed.

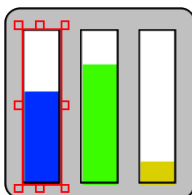
### 7.8.2.7. Flow Indicator



**Figure 7.24. Flow Indicator Beans**

The flow indicator bean is used to indicate flow beyond a forward and reverse threshold pair. When the data value is above the forward threshold the indicator will display an up arrow. When the data value is below the reverse threshold the indicator will display a down arrow. When the data value is between the two thresholds the indicator will be displayed as a single line. The flow indicator can be customized with a foreground color for forward and backward indication, a border and an orientation (north, south, east or west).

### 7.8.2.8. Fluid Level



**Figure 7.25. Fluid Level Bean**

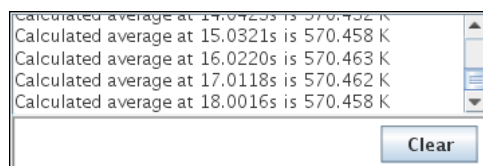
The Fluid Level bean is a specialized version of the Single Volume bean. This bean has a *Level Data Channel* that is used to indicate the liquid fluid level (using specified minimum and

maximum values) and a *Volume ID* used to determine the color of the filled region. Like many other display beans, this bean includes a background color, border, tooltip text, etc.

### 7.8.2.9. Polygon

The Polygon is the same object as the Polygon Annotation described in [the section called “Polygon”](#). When placed in an animation model the polygon includes a *Data Source*, *Range* and an either a *Data Channel* or a *Volume ID*.

### 7.8.2.10. Python Output Display



**Figure 7.26. Python Output Bean**

The Python Output bean is a specialized bean for displaying the printed output and error streams of the Python Data Source ([Section 7.5, “The Python Data Source”](#)). The font, background color, border, and error message display properties can be used to customize this bean.

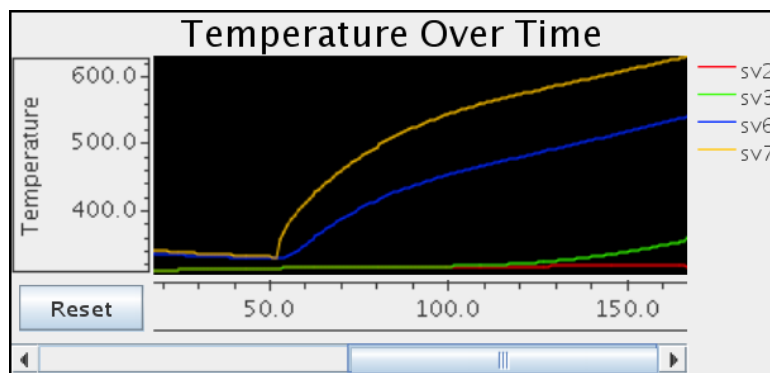
### 7.8.2.11. Simple Annunciator



**Figure 7.27. Simple Annunciator Bean**

The Simple Annunciator bean is used to indicate 'On' when the data value has increased beyond the specified *On Threshold* and then 'Off' when the value has decreased beyond the specified *Off Threshold*. The text displayed ('On' above) can be specified by changing the *On Text* property. When off, no text is displayed. Also, like many other display beans, this bean includes a background color, border, tooltip text, etc.

### 7.8.2.12. Strip Plot



**Figure 7.28. Strip Plot Bean**

The Strip Plot bean is a color coded line plot of multiple data channels. A Strip Plot requires a single *Time Data Channel* and at least one data channel (*Data Channel 1* through *Data Channel 5*) specified in order to produce a plot. A single plot point is added to the line for each timestep animated. Pressing the **Reset** button or rewinding time will clear all plotted points and restart the plot from the new current time.

**Note** The **Reset** button and time scrollbar can only be used when the 2D View is locked.

In addition to the normal font, background color, etc. this bean can be customized by setting the following plot appearance related properties.

- *Plot Background Color* - The background color of the plot region.
- *Line Thickness* - The thickness (in pixels) of the plotted line.
- *Show Legend* - When set to *True* the legend (shown at the top right) will be displayed showing the channel names and their corresponding line colors.
- *Show Scrollbar* - When set to *True* the scrollbar will be displayed at the bottom of the plot to allow scrolling to previously plotted data.
- *Y-Axis Label* - This property is the label used to describe the y-axis shown as *Temperature* above.
- *Show Units* - When set to *True* the units of the y-axis will be appended to the Y-Axis label. Thus, the above *Temperature* would be displayed as *Temperature K*.
- *Time Base* - This property determines the x-scaling of the plot and how much data is visible at one time.

## 7.8.3. Interactive

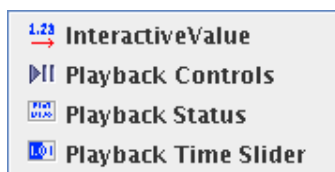


Figure 7.29. Interactive Beans

### 7.8.3.1. Interactive Value

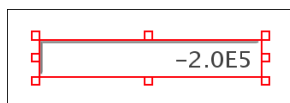


Figure 7.30. Interactive Value Bean

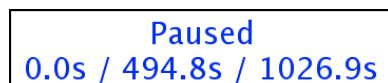
This bean is designed as a controller for an interactive variable inside a calculation. In essence, it is an enhanced Data Value that allows the retrieved value to be edited and sent as an interactive command. To use this bean, first specify a *Data Channel* to be displayed. Then, specify the

*Variable Name* of the interactive variable in the calculation. During animation the data value will be displayed in red for a brief time each time the value changes.

### 7.8.3.2. Playback Controls

The Playback Controls bean is identical to the controls available on the main toolbar. Refer to [Section 7.7, “Animation Playback”](#) for more information on these controls.

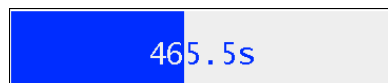
### 7.8.3.3. Playback Status



*Figure 7.31. Playback Status Bean*

The playback status bean shows the status of the current animation including the current, start and end times. The time display can be customized with a numerical format as well as the standard font and foreground color properties.

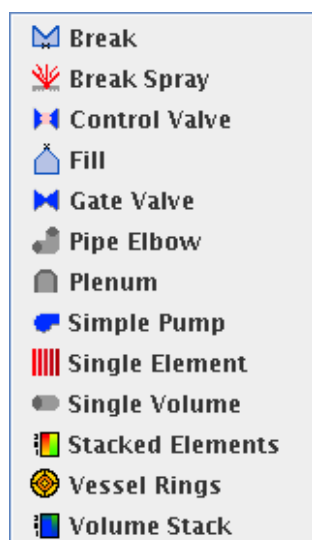
### 7.8.3.4. Playback Time Slider



*Figure 7.32. Playback Time Slider Bean*

This bean can be used to quickly scroll through to an approximate location in a calculation. Clicking inside the bounds (while the view is locked) will skip the playback to a time relative to the clicked position in the slider. Clicking and dragging to interactively skip the animation following the mouse. The time display can be customized using a numerical format, font and foreground color or hidden entirely.

## 7.8.4. Plant Components



*Figure 7.33. Plant Component Beans*

### 7.8.4.1. Break



**Figure 7.34. Break Bean**

The Break bean is a simple indication of whether a data value is above a specified *Threshold Value*. The break display can be customized by its *On Color*, *Off Color*, *Orientation* and *Line Width*.

### 7.8.4.2. Fill



**Figure 7.35. Fill Bean**

This bean differs from the Break Bean only in its shape.

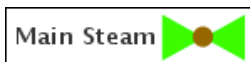
### 7.8.4.3. Break Spray



**Figure 7.36. Break Spray Bean**

The Break Spray is intended to represent a break in a fluid system with a color change and a spraying animation. The properties available for this bean are nearly identical to the Break Bean with the exception of the *Line Width* used to control the thickness of the spray lines.

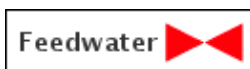
### 7.8.4.4. Control Valve



**Figure 7.37. Control Valve Bean**

The Control Valve bean represents a valve that can be partially open. When the data value is greater than or equal to the *Open Value* the valve is considered open and is drawn using the *Open Color*. When the data value is less than or equal to the *Closed Value* it is considered closed and is drawn using the *Closed Color*. Values in between the two are considered partially open and an interpolated color between the two colors is used.

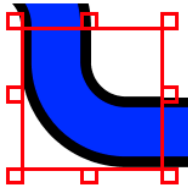
### 7.8.4.5. Gate Valve



**Figure 7.38. Gate Valve Bean**

The Gate Valve bean represents a valve that can only be fully open or fully closed. It differs from the Break Bean only in its shape.

#### 7.8.4.6. Pipe Elbow



**Figure 7.39. Pipe Elbow Bean**

The pipe elbow bean is a more customized version of a Single Volume bean that is intended to resemble a piping elbow. Like the Single Volume, the this bean uses a *Volume ID* to select the data to animate. In addition to the normal Font, Foreground Color, Border, etc. this bean's shape and appearance can be customized with the following properties.

- *Curved Fraction* - The fraction of the segment length that is curved. Smaller values result in a sharper turn. ( 0.0 - 1.0 )
- *Pipe Width Fraction* - The thickness of the pipe segment in relation to the width of the elbow. Smaller values result in a thinner pipe. (0.0 - 1.0)
- *Orientation* - The direction the inner curve is pointed: North-East, North-West, South-East or South-West.
- *Outline Width* - The thickness of the outline (in pixels).

#### 7.8.4.7. Plenum



**Figure 7.40. Plenum Bean**

The plenum bean is a more customized version of a Single Volume bean that is intended to resemble an upper or lower plenum. Like the Single Volume, the this bean uses a *Volume ID* to select the data to animate. In addition to the normal Font, Foreground Color, Border, etc. this bean's shape and appearance can be customized with the following properties.

- *Plenum Type* - The plenum orientation type. (Upper or Lower)
- *Curved Fraction* - The fraction of the plenum that is curved. ( 0.0 - 1.0 )
- *Outline Width* - The thickness of the outline (in pixels).

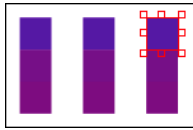
#### 7.8.4.8. Simple Pump



**Figure 7.41. Simple Pump Bean**

The Simple Pump bean is a simple indication of whether a data value is above a specified *Threshold Value*. The pump display can be customized by its *On Color*, *Off Color* and *Facing East*.

### 7.8.4.9. Single Element



**Figure 7.42. Single Element Bean**

The Single Element bean displays a color based on the data value from the specified *Data Channel* using the specified *Range*. This bean can be either a rectangle or an ellipse based on *Element Shape* property.

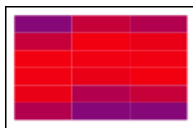
### 7.8.4.10. Single Volume



**Figure 7.43. Single Volume Bean**

The Single Volume bean displays a color based on the specified *Volume ID* and *Range*. This bean can be used to represent a vertical or horizontal pipe segment by setting the *Outline Width* (set to 3 above) and the *Orientation*. This bean can also be rectangular or elliptical based on the *Volume Shape* property.

### 7.8.4.11. Stacked Elements



**Figure 7.44. Stacked Element Bean**

The Stacked Element bean is a 2 dimensional stack of Single Element beans designed to simplify the process of visualizing areas of dense nodalization or long stretches of sequential channel numbering. The example shown in [Figure 7.44](#), “Stacked Element Bean” has 6 rows and 3 columns of fuel rod temperatures.

To create a complete Stacked Element bean:

1. Insert a new Stacked Element bean using the Insert Tool.
2. Set the number of columns in the bean by adding entries to the *Column Widths* table.

This table is used (in normalized form) to scale the size of each column in the bean. The number of entries in this table determines the number of columns and the value for each determines its relative width.
3. Set the number of rows in the bean by adding entries to the *Row Heights* table.



This table, like the Column Widths table, is used to determine the number of rows and their relative heights.

4. Specify data channel names for each element in the stack.

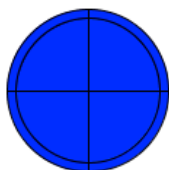
Open the *Select Channels* dialog by editing the *Node Volumes* property of the bean. The table at the top of this dialog contains the channel names for each element in the bean layed out by row and column in the same order as they will be rendered by the bean. These channels can be input individually (by editing each table cell), selected individually (by selecting the **Channel Selection** radio button) or input in groups by specifying a **Pattern**.

Refer to the pop-up help of the Select Channels dialog for a more detailed discussion of channel selection individually and by pattern.

In addition to the normal bean customization, the stacked element bean provides 2 stack-specific properties:

1. *Opaque Nodes* - When this is set to False, nodes that do not have channels specified will be transparent.
2. *Draw Outline* - When this is set to true, a line border will be drawn in the foreground color around each element in the stack and around the entire stack.

### 7.8.4.12. Vessel Rings



*Figure 7.45. Vessel Rings Bean*

The Vessel Rings Bean is a series of concentric circles each divided into azimuthal sectors. The Rings may have any number of rings, and each ring may have any number of azimuthal sectors. Each azimuthal sector is assigned a volume number, and the first sector may include an offset angle indicating where it begins around the circle. Optionally, regions that have no volume number may be transparent, appearing as a void in the volume.

### 7.8.4.13. Volume Stack

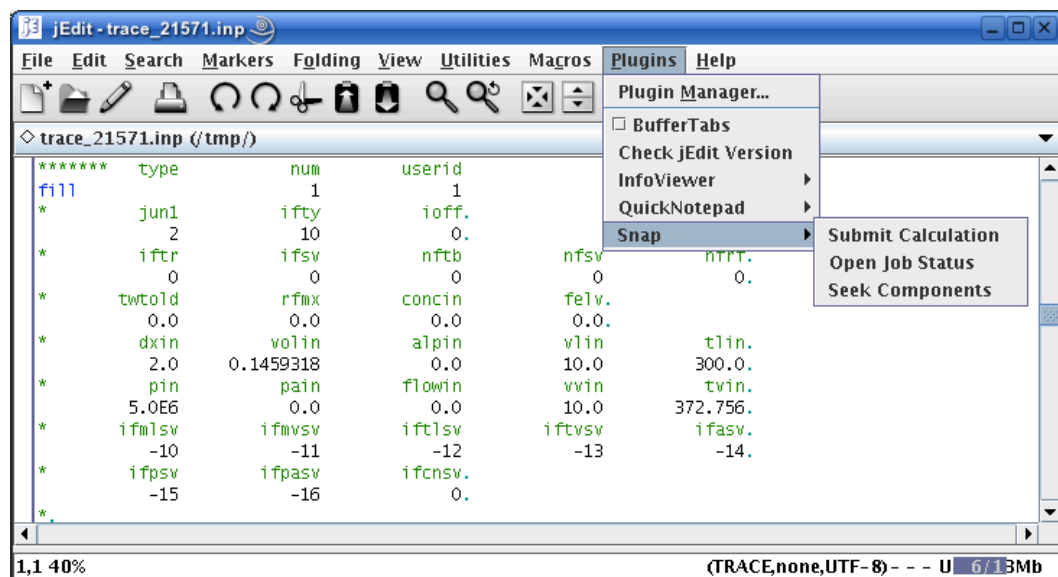
This bean is a 2 dimensional stack of Single Volume beans. This bean differs from the Stacked Element only in that *Volume IDs* are selected for each element rather than channel names.

---

# Chapter 8. Using the jEdit Plug-in

jEdit is a pure-Java, programmer's text editor that was chosen for use with the SNAP system because of its stability and multitude of features. The built-in support for syntax highlighting definitions and extensible plug-in architecture make it a very valuable tool when working with ASCII formatted input files. For a detailed listing of jEdit's available features see the jEdit homepage at <http://jedit.sourceforge.net>. For detailed instructions on installing the SNAP jEdit plug-in refer to [Appendix A, Installing the jEdit Plug-in for SNAP](#).

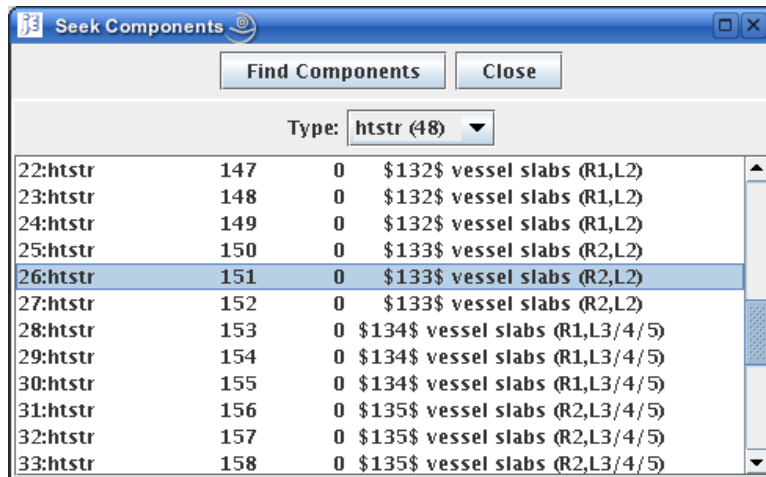
The jEdit editor is used primarily for the Export to jEdit feature available in most SNAP plug-ins. This feature is used to export the ASCII input for the selected object (model, component, etc.) to a temporary file which is then opened in jEdit. There, the exported ASCII can be reviewed or copied and pasted into other input files.



**Figure 8.1. The jEdit Editor**

The SNAP jEdit Plug-in provides the following additional features to jEdit:

- **Job Submission** - The Submit Calculation menu item will open the submit dialog for use in submitting the currently open file to a Calculation Server. Refer to [Chapter 5, Submitting a Job](#) for more information on the job submission process.
- **Syntax Highlighting** - The jEdit plug-in currently supports syntax highlighting for the TRACE and RELAP5 analysis codes.
- **Seek Components** - The Seek Components dialog (shown in [Figure 8.2, “The Seek Components Dialog”](#)) is used to locate components within the input deck. The type dropdown contains a list of component types found in the model along with the number of each type located in parentheses. The components for the selected type are displayed in the list. Selecting a component will result in that component being scrolled into view. The Seek Components dialog currently supports the TRACE and RELAP5 analysis codes.



*Figure 8.2. The Seek Components Dialog*

---

# Appendix A. Installing the jEdit Plug-in for SNAP

The following steps are required to install and configure jEdit to work with SNAP:

1. [Installing jEdit](#)
2. [Installing the SNAP jEdit Plug-in](#)
3. [Update the jEdit Configuration](#)
4. [Set the SNAP Installation Directory](#)
5. [Section A.5, “Set the jEdit Executable”](#)

## A.1. Installing jEdit

The current jEdit plug-in for SNAP is designed to work with jEdit version 4.1. Other versions are not supported and may not work. Platform specific instructions on installing jEdit are available in the Download section of the jEdit homepage: <http://jedit.sourceforge.net>.

An installer and user's manual for jEdit 4.1 is included as an optional package in the SNAP distribution. If you selected this option during SNAP installation or upgrade, the following files should have been placed in a `jedit` subdirectory under your SNAP installation directory:

- `jedit41install.jar` - The jEdit 4.1 installer.
- `jedit41manual-a4.pdf` - The jEdit 4.1 user's manual.

Start the jEdit 4.1 installer under Java and follow the instructions to complete installation.

## A.2. Installing the Plug-in Files

The SNAP plug-in for jEdit 4.1 is also included as an optional package in the SNAP distribution. If you selected this option during SNAP installation or upgrade, the `jars` and `modes` folders should have been placed in a `jedit` sub-folder in your SNAP installation folder. These folders contain the SNAP plug-in for jEdit and the SNAP syntax highlighting modes respectively.

1. Copy the file `<installfolder>/jars/snap.jar` to the `jars` folder under the jEdit install folder.
2. Copy all of the `*.xml` files from the `<installfolder>/modes/` folder to the `modes` under the jEdit install folder.

## A.3. Update the jEdit Configuration

- **Under Windows** - Start the jEdit configuration tool, `jeditinit`, located in the jEdit installation folder and add the following to the *Command Line Options for Java Application Loader* (on one line):

```
-mx32m -Djava.endorsed.dirs="<installfolder>\\lib\\corba" -jar
```

Where <installfolder> is your SNAP installation folder.

- **Under Unix** - Find the script that starts jEdit called `jedit`. Edit the line that begins with **exec** and edit it so that it looks like (on one line):

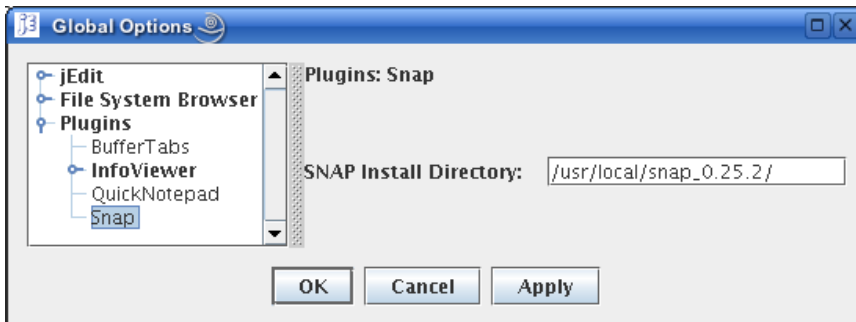
```
exec      <path      to      java>/java      -mx32m      -  
Djava.endorsed.dirs="<installfolder>/lib/corba"  ${JEDIT}  -jar  
"<jeditinstalldir>/lib/jedit.jar"  $@
```

Where <installfolder> is your SNAP installation folder, <path to java> is the path to your java executable, and <jeditinstallfolder> is your jEdit installation folder. The entire command should be on one line.

## A.4. Set the SNAP Installation Directory

Start jEdit and enter the SNAP installation directory when prompted. If the SNAP jEdit plug-in was already installed, this setting can be updated by opening the Global Options dialog from the **Utilities -> Global Options** menu item.

**Note** jEdit must be restarted for this change to take effect.



*Figure A.1. Snap Installation Directory Option*

## A.5. Set the jEdit Executable

Set the *jEdit Executable* preference in the Configuration Tool [Section 3.2.1, “General Properties”](#) section to refer to the installed jEdit executable.

---

# Appendix B. Calculation Job Files

The Calculation Job File (CJF) format (\*.cjf) replaced the previous .status file format used by the Calculation Server. Information that was stored in a job's .status file is now stored in the job's CJF. See [Section B.1, “A Sample Calculation Job File”](#) for a more detailed explanation of the CJF format.

The CJFs used by the Calculation Server are created by the Server Plug-in (TRACE, RELAP5, etc.) for each job that is added. In some cases an analysis code may require that its input/output files have pre-defined names. For these situations a sub-folder will be used to contain these jobs and the CJF will be placed in the parent folder. Any number of jobs can coexist in a single on-disk folder or sub-folder. This feature could be used by an automated test suite to allow immediate access to the suite's results by simply adding the suite's folder and creating a \*.cjf file for each job.

All CJF files include the following data:

- Job Name
- Plug-in ID
- Prerequisite Jobs (if any)
- Restart Jobs (if any) – Jobs whose restart data was (or will be) used by this job.
- File Names – All files related to this job including input, output, etc.
- Calculation start and end times (if available)
- Additional plug-in specific information stored by the server plug-in managing the job. (TRACE, RELAP5, etc.)

Calculation Job Files are scanned by the Calculation Server only when requested by a client application. This on-demand scanning allows the Calculation Server to manage a much larger pool of completed jobs with a minimum of memory and processor usage.

## B.1. A Sample Calculation Job File

The following is a sample Calculation Job File created for a TRACE job named TestJob. The CJF files for jobs using other plug-ins will differ only in the FILE entry names used. Note that the order of the entries in a CJF is not important but they are case sensitive.

In the following example, the header (!CJF) is required and must be in the format shown. The format of this header is used to determine if the CJF is a valid file. Comments start with # and are completely optional. Dates (SUBMITTED, STARTED and COMPLETED) are stored as a UNIX timestamp. Times (START\_TIME, END\_TIME and CALC\_TIME) are stored in seconds. Files ( [ { FILE } ]\_\* ) are stored as relative file locations when possible.

```
#!CJF# C:\Runs\TestJob.cjf
#Fri Sep 01 09:26:55 EDT 2006
NAME=TestJob
PLUGIN=TRACE
EXECID=V4260MOD
PRIORITY=5
```

```
START_TIME=0.0
END_TIME=588.392578125
CALC_TIME=588.392578125
SUBMITTED=1156937862323
STARTED=1156937865365
COMPLETED=1156938057558
[{FILE}]_DEMUX=TestJob.dmx
[{FILE}]_INLAB=TestJob.lab
[{FILE}]_INPUT=TestJob.inp
[{FILE}]_MED=TestJob.med
[{FILE}]_MESSAGE_IDX=TestJob.msg.idx
[{FILE}]_MESSAGE=TestJob.msg
[{FILE}]_MUX_CACHE=TestJob.xtv_cache
[{FILE}]_MUX=TestJob.xtv
[{FILE}]_OUTPUT_IDX=TestJob.out.idx
[{FILE}]_OUTPUT_RDX=TestJob.out.rdx
[{FILE}]_OUTPUT=TestJob.out
[{FILE}]_RESTART_IN=TestJob.rst
[{FILE}]_RESTART=TestJob.tpr
[{FILE}]_RST_DUMP=TestJob.dmp
[{FILE}]_SCREEN_IDX=TestJob.screen.idx
[{FILE}]_SCREEN=TestJob.screen
[{FILE}]_TRACEIN=TestJob.tsp
[{FILE}]_TRCDIF=TestJob.dif
[{FILE}]_TRCEXT=TestJob.extr
[{FILE}]_TRCH2O=TestJob.h2o
[{FILE}]_TRCINP=TestJob.echo
[{FILE}]_TRCSNI=TestJob.sni
[{FILE}]_TRCSNO=TestJob.sno
[{FILE}]_TRCSTATS=TestJob.stats
```

CJFs created by external applications must include the following:

- **#!CJF** - The header in the correct format.
- **NAME** - The name of the job including only letters, numbers, underscores and hyphens.
- **PLUGIN** - The plug-in ID of the plug-in that will be used to access this job. (e.g. RELAP5, TRACE, COBRA, EXTDATA)
- **[{FILE}]\_MUX** - The multiplexed data file used by this job.



---

# Index

## A

analysis codes, 1

## C

Calculation Server

- demux executable, 40

- executables, 39

check model

- main menu, 4

checking for updates, 36

configuration

- Calculation Server, 37

- general properties, 36

- saving, 36

- web support, 40

## F

file operations

- main menu, 4

- main toolbars, 4

## J

jEdit

- export from main menu, 4

## M

main property view, 6

- attribute groups, 6

- component selection button, 7

- custom editor button, 7

- disabled check box, 7

- help button, 6

memory toolbar, 4

models

- name, 5

## N

navigator, 5

- plug-in node, 5

- pop-up menus, 5

## P

properties, 6

## S

server

- adding, 43

- definition of, 51

- local, 51

- remote, 51

- removing, 43

server keys

- exporting, 36

- generating, 36

Symbol Nuclear Analysis Package, 1

## U

undo/redo

- main menu, 4

- main toolbars, 4

user interface, 3

- main toolbars, 4

- view toolbars, 8

## W

window, 5

---